Evolutionary Intelligence
https://doi.org/10.1007/s12065-023-00836-0

SPECIAL ISSUE

Highly adaptive regression trees

Sohail Nizam[1] · **David Benkeser**[1]

Received: 4 November 2022 / Revised: 23 January 2023 / Accepted: 24 January 2023

Abstract
The development of machine learning methods that are both accurate and interpretable is of paramount importance in healthcare and many other fields. The Highly Adaptive Lasso (HAL) has been shown to have predictive performance on par with state-of-the art algorithms. HAL involves performing regularized regression of the outcome on a tensor product of indicator basis functions. In this paper we show that this basis can be represented as a non-recursive partitioning of the feature space and propose a method for mapping this partitioning implied by HAL to a recursive partitioning. Such a mapping then allows for the representation of HAL as a decision tree, thereby providing interpretability of predictions made by the algorithm. We refer to this post-hoc method for interpretability as Highly Adaptive Regression Trees (HART). We provide a set of algorithms to construct this mapping and conveniently visualize the resulting tree. Using real data, we show that HAL's predictive performance is on par with state-of-the-art methods, and we demonstrate the construction and interpretation of HARTs.

Keywords Machine learning · Interpretability · Decision trees · Recursive partitioning · Highly adaptive lasso

1 Introduction

Recent advances in theoretical statistics and computer science have led to the development of machine learning algorithms that can improve professionals' abilities to accomplish tasks in a myriad of fields. For example, such algorithms have been used to assess risks of future adverse health outcomes [1–4], estimate consumer demand [5], forecast currency exchange rates [6], predict student academic success [7], and improve communication systems [8]. Machine learning may be used by professionals in these fields to appropriately guide decision making. However, use of machine learning in these contexts often involves weighing the choice of accuracy versus interpretability of a prediction algorithm. Simple algorithms (e.g., based on logistic regression) are easy-to-interpret, but rely on relatively inflexible statistical models and so may not make accurate predictions. More complex algorithms (e.g., based on deep learning) may predict more accurately, but are often difficult to interpret.

The need for accuracy in these settings is manifest. For example, before machine learning-based technology can be deployed in healthcare settings, providers and patients must have faith in the fidelity of the predictions. Interpretability is also crucial from an ethical standpoint [9]. Consider an algorithm to triage intensive care unit patients based on predicted risk of death. In this case, it is important to carefully scrutinize the triage decision making process to ensure that care is being delivered in an equitable manner.

Decision trees have long been a popular tool for creating interpretable prediction functions and are particularly popular in clinical research [10–13]. However, traditional techniques for learning these trees generally fail to yield predictions that are as accurate as competitor algorithms [14]. The poor performance may be due to the greedy approach used to fit the trees, which can quickly spread data thin even in relatively data rich settings. This has led to the abandonment of regression trees in settings where accuracy is a primary consideration, in favor of other, more difficult-to-interpret algorithms.

Here we describe a method for constructing classification and regression trees that delivers predictive accuracy comparable to that of random forests and boosting. Our proposal centers around the highly adaptive lasso (HAL) algorithm

✉ Sohail Nizam
sohail.nizam@emory.edu

David Benkeser
benkeser@emory.edu

[1] Department of Biostatistics and Bioinformatics, Rollins School of Public Health, Emory University, 1518 Clifton Road, Atlanta, Georgia 30322, USA

Published online: 14 March 2023

[15], which has been previously established to have competitive predictive performance compared with state-of-the-art algorithms. Here, we show that HAL can be represented as a decision tree and provide an algorithm for building such a tree from a trained HAL model. We verify that HAL can indeed perform prediction tasks as well as state-of-the-art algorithms, and we demonstrate the interpretability of trees built using real data.

2 Highly adaptive lasso

We consider a prediction problem where the observed data consist of n independent copies of the random variable $O = (X, Y) \sim P_0$, where $X \in \mathcal{X}$ is a vector of features, $Y \in \mathcal{Y}$ is the outcome of interest, and P_0 represents the probability distribution of the observed data. Here, $\mathcal{Y}$ could be $\{0, 1\}$ (as in binary classification), $\{\mathbb{Y}_1, \ldots, \mathbb{Y}_k\}$ (multi-class classification), or $\mathbb{R}$ (regression). For simplicity, we take $\mathcal{Y} = \{0, 1\}$.

For a given function $\psi \in \Psi$, the space of functions mapping from $\mathcal{X}$ to the unit interval, we define the negative log-likelihood loss function, $L(\psi, o) = -\log[\psi(x)^y\{1 - \psi(x)\}^{1-y}]$. The *risk* of ψ is the expected value of $L(\psi, O)$, $R_0(\psi) = \int L(\psi, o) dP_0(o)$. It is straightforward to show that the minimizer of this risk over Ψ is ψ_0, the conditional probability that $Y = 1$ given X. The theory of HAL is built around two key assumptions about ψ_0: (i) ψ_0 is right-continuous with left-hand limits (i.e., is a *cadlag* function) and (ii) ψ_0 has finite variation norm. To precisely define variation norm, we note that any cadlag function ψ generates a signed measure, which allows us to write integrals with respect to ψ. The variation norm of ψ is $\|\psi\|_v = \int |d\psi(u)|$.

Previous works [15, 16] demonstrated how to find the minimum loss-based estimator (MLE) in the class of functions with variation norm smaller than a fixed number M. This MLE estimates $\psi_{0,M} = \operatorname{argmin}_{\psi : \|\psi\|_v < M} R_0(\psi)$ and may be computed using the fact that any cadlag function with finite variation norm can be arbitrarily well-approximated by a tensor product of indicator basis functions. In the univariate case, we observe $X_1, \ldots, X_n$, independent copies of a scalar-valued variable X. The linear combination of basis functions is of the form $\psi_{n,\beta}(x) = \beta_0 + \beta_X^\top b(x)$, where $b(x)$ is an n-length vector with i-th entry equal to $\mathbb{1}_{[X_i,\infty)}(x)$. Here, we use the indicator notation: for $(c, d) \in \mathbb{R}^2$, $\mathbb{1}_{[c,d)}(x) = 1$ if $x \in [c, d)$ and equals 0 otherwise. As the dimension of X increases, we include tensor product basis functions. For example, if we observe $(X_{1i}, X_{2i}), i = 1, \ldots, n$, independent copies of $X = (X_1, X_2)$, where $X_1, X_2 \in \mathbb{R}^2$, then $\psi_{n,\beta}(x) = \beta_0 + \beta_{X_1}^\top b_1(x) + \beta_{X_2}^\top b_2(x) + \beta_{X_1,X_2}^\top b_1(x) b_2(x)$, where for $j = 1, 2$, the i-th entry of $b_i(x)$ is equal to $\mathbb{1}_{[X_{ji},\infty)}(x_j)$. The idea generalizes to arbitrary p with at most $n(2^p - 1)$ basis functions included.

Because the L_1-norm of β equals the variation norm of $\psi_{n,\beta}$, the MLE in the class of functions with variation norm smaller than M may be computed by regressing Y onto this set of basis functions, and ensuring that the $\sum_{i=1}^n |\beta_i| < M$. This can be achieved by making use of software for the popular lasso algorithm [17]. We note that with a univariate feature, this approach is well established in the signal de-noising literature under the name of (zero-order) trend filtering. Benkeser and van der Laan (2016) [15] generalized to multiple dimensions and proved that HAL converges to ψ_0 in terms of regret at a fast rate, $R_0(\psi_{n,M}) - R_0(\psi_{0,M}) = o_p(n^{-[1/4+1/\{8p+1\}]})$. This theory establishes HAL as a strong candidate for accurate machine learning. In Sect. 4, we establish that HAL is also a good candidate for performing interpretable machine learning.

3 Notation

The remainder of the paper includes a large set of notation that is necessary for describing the process of building tree representations of HAL models. For this reason, we include Table 1 which contains descriptions of all new notation and vocabulary introduced. Each piece of notation is defined in the text, but readers my find it convenient to refer to all definitions in one place.

4 From HAL to HART

Classification and Regression Tree (CART) [18] is a prediction algorithm that has long been employed across many diverse fields [10]. Decision trees built using this approach typically start with a fully un-partitioned feature space and, through a greedy algorithm, recursively partition the (sub) space into smaller units until a specified stopping criterion is met. By nature of the partitioning process, CART yields a histogram approximation of ψ_0. Furthermore, because of the recursive process employed in training, decision trees are naturally created. However, CARTs rarely achieve high predictive performance and can be prone to overfitting [14].

HAL provides an alternative approach to learning a histogram approximation of a function, though its training process is in stark contrast to CART. HAL starts with a dense partitioning of the feature space (implied by the tensor product basis expansion), that is subsequently shrunk to achieve the best global fit to the data. Because HAL is optimizing a global smoothness criteria, it is able to partition the feature space more efficiently than CART and generally provides a much more accurate representation of ψ_0. However, the histogram approximation created by HAL partitions will not in

Table 1 Notation used throughout the paper to describe aspects of feature space partitioning and tree creation

Term	Description
$\mathbb{X}$	Candidate split data structure to be recursively pruned
$\mathcal{R}$	A feature space region
$b_{ji}(x_j)$	Indicator basis function set to 1 if $X_j \geq x_{ji}$
subscript G	A label applied to indicate right-hand sub-region
subscript L	A label applied to indicate left-hand sub-region
Φ_{ji}	Set of basis expansion terms involving value X_{ji}
Φ	Set of all Φ_{ji} (represents the whole HAL model fit)
π	Policy for choosing tree split
Tree	A HART tree object
Tree.label	Text displayed in the rood node of Tree
Tree.preds	Set of predictions given by terminal nodes in Tree
Node(label)	Indicates creation of a non-terminal node displaying label
TerminalNode	Indicates creation of a terminal node displaying label
Tree.left-child	Left child node of the root node of Tree
Tree.right-child	Right child node of the root node of Tree
subtree.parent = node	Indicates assignment of node as parent of object subtree

general represent a recursive partitioning of the feature space and thus will not be immediately amenable to representation as a decision tree. The goal of this paper is to present an algorithm that maps a trained HAL model into a decision tree. We call the resulting trees Highly Adaptive Regression Trees (HART).

To illustrate the significant challenges associated with this goal, we consider an immensely simplified example in which we observe $(X_{1i}, X_{2i}), i = 1, 2, n = 2$ observations of the two-dimensional feature-vector $X = (X_1, X_2)$. Further assume that $X_{1i} < X_{2i}$ for $i = 1, 2$. Define $b_{ji}(x) = \mathbb{1}_{[X_{ji},\infty)}(x_j)$. The HAL basis expansion is

$$\psi_{n,\beta}(x) = \beta_0 + \beta_{X_{11}} b_{11}(x_1) + \beta_{X_{12}} b_{12}(x_1) + \beta_{X_{21}} b_{21}(x_2) + \beta_{X_{22}} b_{22}(x_2) + \beta_{X_{11},X_{21}} b_{11}(x_1) b_{12}(x_2) + \beta_{X_{12},X_{22}} b_{12}(x_1) b_{22}(x_2) \quad (1)$$

The left panel of Fig. 1 illustrates the partitioning of the feature space implied by this basis expansion. For example, given $\hat{\beta} \in \mathbb{R}^7$, the HAL prediction for observations falling in the lowest left portion of the feature space is $\hat{\beta}_0$, for the upper left portion the prediction is $\hat{\beta}_0 + \hat{\beta}_{X_{21}} + \hat{\beta}_{X_{22}}$, and so on. Note that the full partitioning implied by the basis expansion (equivalent to the partitioning implied if all coefficient estimates are nonzero) is naturally recursive and thus can immediately be represented by the decision tree in Fig. 2. In this and subsequent trees, each node contains an observed value; moving to the left (right) of the node implies a value less (equal to or greater) than the node value.

Because HAL uses L_1-penalization of the coefficient vector, many coefficients are shrunk to zero when the model is trained; this causes some partitions of the feature space to collapse. For example, suppose we find that

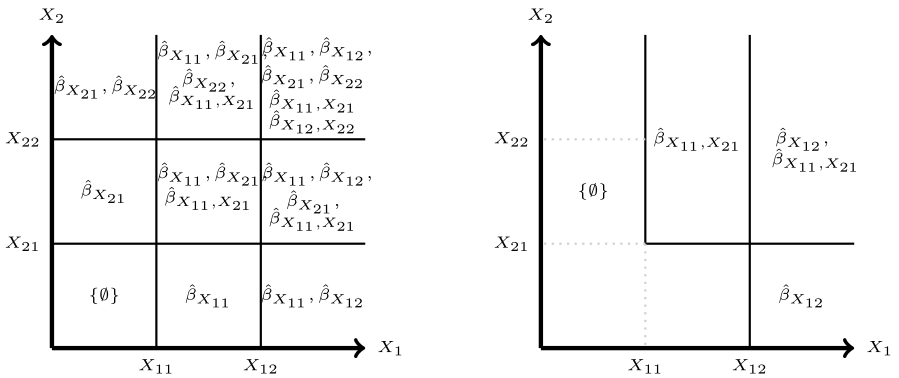

Fig. 1 Left: Illustration of partitioning of the feature space according to the HAL basis expansion. The sets of $\hat{\beta}$ (excluding the intercept) correspond to those in equation (1) that fall in each region. Right: An illustrative HAL partitioning, when $\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{21}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{12},X_{22}} = 0$, which results in a non-recursive partitioning of the feature space. To represent this partitioning as a decision tree, we must determine a parsimonious recursive structure for the L-shaped left-most region

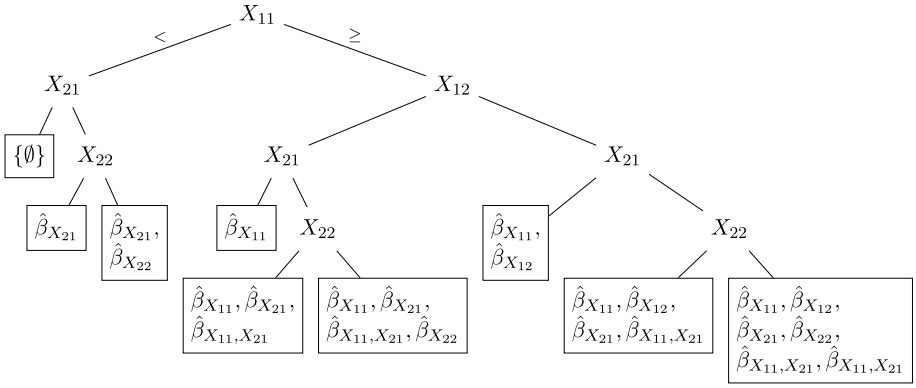

Fig. 2 Decision tree representation of the full feature space partitioning implied by the HAL basis expansion

$\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{21}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{12},X_{22}} = 0$. In this case, the partitioning collapses (right panel, Fig. 1) and *does not have a recursive structure*. In this case, any decision tree representing this partitioned space will have at least some redundancy in the terminal nodes. Figure 3 illustrates two possible recursive partitions that could be used to approximate the non-recursive partitioning implied by HAL. Note that these are both valid representations in that they would yield predictions identical to those generated by the HAL model. However, each has one redundant partition. This can be observed by noting that each has separate regions with identical predictions.

In higher dimensions, the partitions cannot be visualized, and some recursive partitions may have a large amount of redundancy. In the next sections we motivate and describe our algorithm for mapping an arbitrary set of non-zero basis functions into a recursive partitioning of the feature space.

Throughout this work, we consider the set $\mathbb{X} = \{X_{ji} : j = 1, \ldots, p \; i = 1, \ldots, n\}$ which contains the values of our data. The elements of $\mathbb{X}$ act as candidates for describing partitions implied by HAL. At each stage in the proposed process, a single value is chosen from $\mathbb{X}$ to describe a partition, and other values are pruned if they are not needed to describe the remaining partitions implied by HAL. The crux of our problem is then identifying which values from our data should be candidates for describing partitions and the order in which we should choose them. As we will see, both of these factors impact the amount of redundancy contained in our final recursive partitioning.

5 Examples

To motivate our approach for reducing redundancy in the HAL tree growing process, we consider three scenarios: (i) a set of basis functions that naturally yields a recursive structure; (ii) a set of basis functions that yields a non-recursive structure for which there are many minimally redundant recursive approximations; (iii) a set of basis functions that yields a non-recursive partitioning for which there are both minimally redundant and highly redundant recursive representations. With each increase in complexity, we will highlight necessary algorithmic changes to the tree growing process.

5.1 Scenario (i): naturally recursive partitioning

Consider a situation where estimates of the coefficients in the basis expansion in equation (1) are all nonzero. The partitioning implied by that model is the full partitioning of the feature space (left panel of Fig. 1). The data structure in our recursive process is $\mathbb{X} = \{X_{11}, X_{12}, X_{21}, X_{22}\}$, the elements

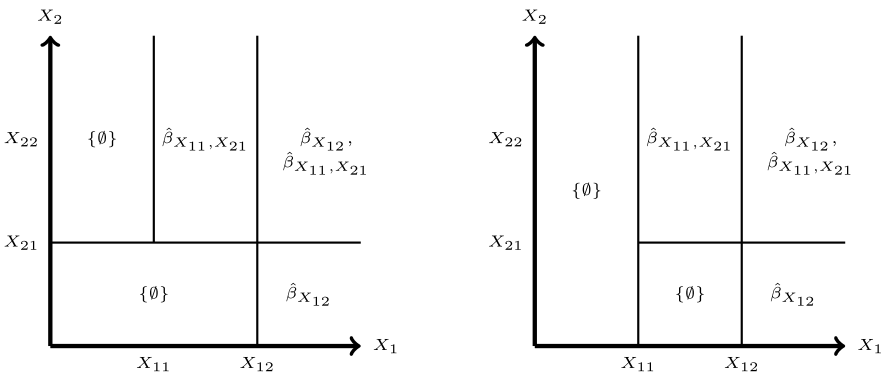

Fig. 3 Two possible recursive partitions to represent the non-recursive partition implied by the HAL model when $\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{21}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{12},X_{22}} = 0$

of which represent potential splits in our feature space and potential nodes in our tree. We can select values from $\mathbb{X}$ to describe feature space partitions in any order. Suppose we begin with value X_{11}. A line is then drawn dividing the feature space a region $\mathcal{R}_L$ where $x_1 < X_{11}$ and a region $\mathcal{R}_G$ where $x_1 \geq X_{11}$ (Fig. 4).

In each of the two new sub-regions of the feature space, we can begin the process again, now with a reduced data structure. In both sub-regions, we set $\mathbb{X} = \mathbb{X}\backslash\{X_{11}\}$ since X_{11} has just been partitioned on. Recall now that we specified $X_{11} < X_{12}$. In sub-region $\mathcal{R}_\text{L}$ where $x_1 < X_{11}$, it must be true that $x_1 < X_{12}$. Thus, it is clearly not necessary to further partition $\mathcal{R}_L$ using the value X_{22}, and we can set $\mathbb{X}_\text{L} = \mathbb{X}\backslash\{X_{12}\} = \{X_{21}, X_{22}\}$. By contrast, in $\mathcal{R}_\text{G}$, $x_1 \geq X_{11}$, but it may or may not be true that $x_1 \geq X_{12}$. Thus, we set $\mathbb{X}_\text{G} = \{X_{12}, X_{21}, X_{22}\}$.

Suppose we carry out this recursive process until $\mathbb{X}$ is empty in every sub-region. Predictions are then constructed by summing the appropriate coefficient estimates for that region. The resulting partitioned feature space and the corresponding decision tree perfectly represent the fully partitioned feature space in the left panel of Fig. 1 and its decision tree in Fig. 2 respectively.

5.2 Scenario (ii): non-recursive partitioning, minimal redundancy

Suppose we estimate $\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{21}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{12},X_{22}} = 0$ and the remaining parameters in 1 to be nonzero. The partitioning implied by this model is pictured in the right panel of Fig. 1. It is worth noting that the full partitioning in the left panel of Fig. 1 is a valid representation of this non-recursive partitioning. However, the associated decision tree would have *considerable* redundancy. Consider two examples of redundancy in that partition under this HAL fit.

First we can see that, X_{22} is not needed to describe any of the partitions implied by the HAL fit. This suggests that our initial data structure $\mathbb{X}$ need not contain X_{22}; we could set $\mathbb{X} = \{X_{11}, X_{12}, X_{21}\}$. Suppose again that we choose X_{11} to describe the first partition. Two sub-regions $\mathcal{R}_\text{L}$ and $\mathcal{R}_\text{G}$ are created where $x_1 < X_{11}$ and $x_1 \geq X_{11}$ respectively. As in Scenario (i), in $\mathcal{R}_\text{G}$ we can simply prune X_{11} and set $\mathbb{X} = \mathbb{X}\backslash\{X_{11}\}$. However, in $\mathcal{R}_\text{L}$, there are no remaining partitions suggesting that in this region X_{11}, X_{12}, and, X_{21} should be pruned from $\mathbb{X}$. This example highlights the necessity for a more comprehensive rule for pruning values from $\mathbb{X}$ in a certain region, given a HAL model fit.

In particular, a value X_{ji} is unnecessary for describing partitions in a given region $\mathcal{R}$ if it meets one of two criteria. First, X_{ji} is unnecessary if all coefficient estimate-basis function product terms involving X_{ji} in the HAL basis expansion evaluate to zero. For example, consider the region where $x_1 < X_{11}$. There, X_{21} is unnecessary because $\hat{\beta}_{X_{21}} b_{21}(x_2) = 0$ and $\hat{\beta}_{X_{11},X_{21}} b_{11}(x_1) b_{21}(x_2) = 0$. The second criterion is that $b_{ji}(x_j) = 1$ for all x_j. For example, X_{11} is unnecessary in the region where $x_1 \geq X_{11}$ because it is known that $b_{11}(x_1) = 1$. Following these criteria, and assuming we select values for partitioning in the order that they appear in $\mathbb{X}$, we obtain the partitioning shown in the left panel of Fig. 3. As we showed in Sect. 4, this is just one possible minimally redundant recursive approximation of the non-recursive partitioning implied by the HAL model. In fact, after initially pruning X_{22}, we could order the elements of $\mathbb{X}$ in six different ways and therefore obtain six different, minimally redundant trees.

5.3 Scenario (iii): non-recursive partitioning, minimal and non-minimal redundancy

Now assume that we estimate $\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{12}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{11},X_{21}} = 0$ and the remaining coefficients as nonzero. The feature space partitioning implied by this model is shown in Fig. 5. According to our criteria from Sect. 5.2, we begin by pruning X_{11} from $\mathbb{X}$. The left and right panels of Fig. 6 show partitionings resulting from proceeding with orderings $\{X_{12}, X_{21}, X_{22}\}$ and $\{X_{21}, X_{22}, X_{12}\}$ respectively. Choosing the first ordering results in an unnecessary partition.

This motivates the introduction of some policy π that dictates which value should be chosen to describe the next

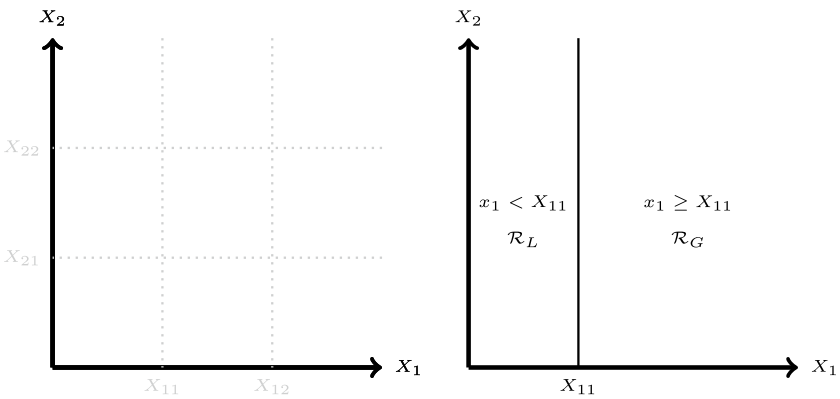

Fig. 4 Left: The feature space partitioned according to the HAL fit with all $\hat{\beta} \neq 0$. These partitions will be described by values from X. Right: Feature space with X_{11} describing the first partition

partition given the data $\mathbb{X}$, the current region $\mathcal{R}$, and the model fit information. The problem of finding the most parsimonious recursive partitioning then reduces to finding an optimal policy π^*. Potential policies are discussed in Sect. 6.2.

6 Highly adaptive regression trees

6.1 Growing trees

Algorithm 1 presents a formal algorithm for finding parsimonious recursive representations of feature space partitionings implied by HAL models. The algorithm is presented for a design matrix involving n observations of p elements, $\mathbb{X} = \{X_{ji} : j = 1, \ldots, p,\ i = 1, \ldots, n\}$. Let Φ_{ji} be the set of all coefficient estimate-basis function product terms involving the value X_{ji} in the HAL basis expansion, and define $\Phi = \{\Phi_{ji} : j = 1, \ldots, p,\ i = 1, \ldots, n\}$, a set of sets that represents the whole HAL model fit. A given region of the feature space can be represented by the tuple $\mathcal{R} = (\Delta, \Gamma)$ where $\Delta(x) = \{X_{ji} : x_j < X_{ji}\}$ and $\Gamma(x) = \{X_{ji} : x_j \geq X_{ji}\}$. At the beginning of the process, we initialize $\Delta = \Gamma = \{\emptyset\}$ and let $\mathcal{R} = (\{\emptyset\}, \{\emptyset\})$ represent the entire feature space.

Once a partition is described using a value X_{ji}, it naturally implies two new regions within the feature space. $\mathcal{R}_{\mathrm{L}}$ describes the region in which $x_j < X_{ji}$ and $\mathcal{R}_{\mathrm{G}}$ describes the region in which $x_j \geq X_{ji}$. These new regions can be formalized by updating the components of the tuple Δ and Γ to include relevant values. As values are added to these sets, basis functions from the HAL basis expansion take on realizations of one or zero thus changing values in Φ. We use the notation $\Phi_{ji} \mid \mathcal{R}$ to refer to the specific realization of a set of product terms Φ_{ji} within a specific region $\mathcal{R}$. Together, the information in the HAL fit and the current region inform which values should be pruned from $\mathbb{X}$ before further partitions are described. We use π to denote a general policy that dictates which value in $\mathbb{X}$ should be chosen to describe the next partition. π is a mapping from the current state of the partitioning, described by $(\mathbb{X}, \Phi, \mathcal{R})$, to the value that will describe the next partition $X_{j^*i^*}$.

Finally, in Algorithm 1 we make the connection between feature space partitions and nodes in a decision tree. Each time we describe a partition using a value X_{ji}, we indicate the creation of a tree node with Node(X_{ji}). The two children

Algorithm 1 HART

$\mathbb{X} = \{X_{ji} : j = 1, \ldots, p,\ i = 1, \ldots, n\}$
$\Phi = \{\Phi_{ji} : j = 1, \ldots, p,\ i = 1, \ldots, n\}$
$\Delta = \Gamma = \{\emptyset\}$
$\mathcal{R} = (\Delta, \Gamma)$
$\mathbb{X} = \mathbb{X} \backslash \{X_{ji} : \Phi_{ji} = \{0, \ldots, 0\} \mid \mathcal{R}\}$
def **Grow**$(\mathbb{X}, \Phi, \mathcal{R})$
if $\mathbb{X} = \{\emptyset\}$ **then**
    $\hat{\psi} = \sum_{s \subset \Gamma} \hat{\beta}_s$
    return TerminalNode$(\hat{\psi})$
else
    $X_{j^*i^*} = \pi(\mathbb{X}, \Phi, \mathcal{R})$
    node = Node$(X_{j^*i^*})$
    $\Delta_{\mathrm{L}} = \Delta \cup \{X_{j^*i} : X_{j^*i} \geq X_{j^*i^*}\}$
    $\mathcal{R}_{\mathrm{L}} = (\Delta_{\mathrm{L}}, \Gamma)$
    $\mathbb{X}_{\mathrm{L}} = \mathbb{X} \backslash \{X_{ji} : \Phi_{ji} = \{0, \ldots, 0\} \mid \mathcal{R}_{\mathrm{L}}\}$
    node.child = **Grow**$(\mathbb{X}_{\mathrm{L}}, \Phi, \mathcal{R}_{\mathrm{L}})$
    $\Gamma_{\mathrm{G}} = \Gamma \cup \{X_{j^*i} : X_{j^*i} \leq X_{j^*i^*}\}$
    $\mathcal{R}_{\mathrm{G}} = (\Delta, \Gamma_{\mathrm{G}})$
    $\mathbb{X}_{\mathrm{G}} = \mathbb{X} \backslash \{X_{ji} : b_{ji} = 1 \mid \mathcal{R}_{\mathrm{G}}\}$
    node.child = **Grow**$(\mathbb{X}_{\mathrm{G}}, \Phi, \mathcal{R}_{\mathrm{G}})$
    return node
end if
call **Grow**$(\mathbb{X}, \Phi, \mathcal{R})$

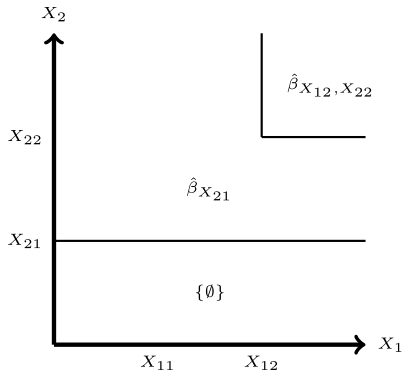

Fig. 5 An illustrative HAL partitioning, when $\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{12}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{11},X_{21}} = 0$, which results in a non-recursive partitioning of the feature space

of that node will correspond to the partitions in the resulting sub-regions where $x_j < X_{ji}$ and $x_j \geq X_{ji}$ respectively.

6.2 Policies

As stated in Sect. 4, the two sources of redundancy in our tree representations are the identification of candidates for partitioning and the order in which we should choose them. Algorithm 1 fully accounts for the former and ensures that no unnecessary feature values are identified as candidates in a given stage of the growing process. The ordering of those candidates and the resulting redundancy is dictated solely by the policy π. Ideally we would choose π to achieve an optimally parsimonious representation of $\hat{\psi}$. However, ordering binary variables to achieve optimally parsimonious representations of functions is known to be an NP-complete problem [19]. We instead rely on heuristics to reduce redundancy.

We propose a heuristic that orders split candidates X_{ji} based on the number of non-zero coefficient estimate-basis function product terms involving X_{ji}. Intuitively, this is the number of times X_{ji} appears in the basis expansion for a given region and can be thought of as a measure of importance. Formally, for split candidates $\mathbb{X}$ and HAL fit Φ in region $\mathcal{R}$ of the feature space, the next split is chosen as

$$\pi_{\text{h}}(\mathbb{X}, \Phi, \mathcal{R}) := X_{j^* i^*} \text{ where } (j^*, i^*) = \text{argmax}_{j,i} |\nu \in \phi_{ji} \mid \mathcal{R} : \nu \neq 0| \tag{2}$$

As an alternative, we propose to take advantage of popular existing decision tree algorithms and define π based on their splitting criteria. This strategy has the benefit of imbuing HART splits with the same interpretation as those of the chosen algorithm as well as acting as a heuristic way of achieving parsimony. For example, in the classification setting, CART chooses splits to minimize Gini Impurity. Heuristically, minimizing Gini Impurity is akin to achieving the 'best separation' of the classes in that region. We can build a HART carrying this same interpretation by employing Algorithm 1 and letting π choose the the minimizer of the Gini Impurity. Formally, if we let $P_{n,\mathcal{R}}$ represent the empirical measure based on data falling in region $\mathcal{R}$, then a Gini Impurity based policy can be defined in the following way. For split candidates $\mathbb{X}$ and HAL fit Φ in region $\mathcal{R}$ of the feature space, the next split is chosen as

$$\begin{aligned}
&\pi_{\text{gini}}(\mathbb{X}, \Phi, \mathcal{R}) \\
&\quad := \underset{x_{ji} \in \mathbb{X}}{\text{argmin}} \{ 2P_{n,\mathcal{R}}(X_j < x_{ji}) P_{n,\mathcal{R}_L}(Y = 1) P_{n,\mathcal{R}_L}(Y = 0) \\
&\qquad + 2P_{n,\mathcal{R}}(X_j \geq x_{ji}) P_{n,\mathcal{R}_G}(Y = 1) P_{n,\mathcal{R}_G}(Y = 0) \}
\end{aligned}$$

We could similarly define π based on the splitting criteria of C4.5 [20], CHAID [21], Minimum Message Length based Decision Graphs [22], or any other algorithm. We can also adapt any of these policies to the needs of the problem setting. Say we are predicting some disease outcome based on treatment status and other features. We could enforce that treatment status is always the final split in the tree and otherwise use π_{gini}. This would allow us to examine how the predicted outcome differs by treatment status for any given subgroup.

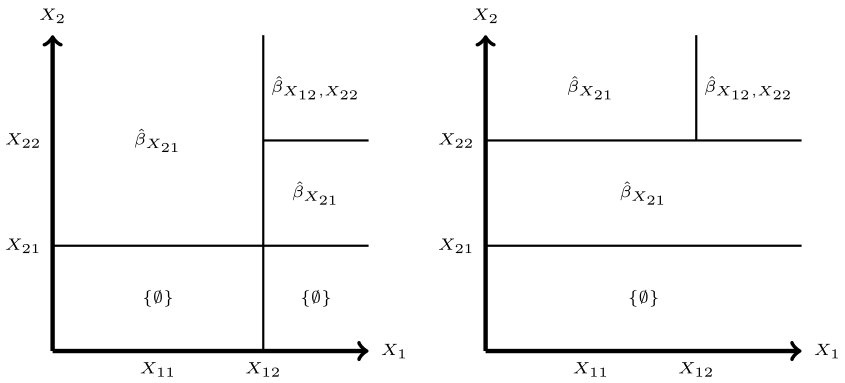

Fig. 6 Two recursive partitionings to represent the non-recursive partitioning implied by HAL when $\hat{\beta}_{X_{11}} = \hat{\beta}_{X_{12}} = \hat{\beta}_{X_{22}} = \hat{\beta}_{X_{11},X_{21}} = 0$. The left partitioning is not minimally redundant. The right partitioning is minimally redundant

6.3 Sub-HARTs and prediction smoothing

Even with a well chosen policy, the full HART may be large and difficult to interpret globally. We can use two simple strategies to mitigate the resultant complexity. First, we can restrict the region of the feature space in which to visualize the model. In Eq. 1, if we restrict $X_1 < X_{11}$, then $b_{X_{11}} = 0$, and we have a simplified function. This simplified function still follows the structure of a HAL model. Therefore Algorithm 1 can be applied to build a Sub-HART that represents the model only in the region where $X_1 < X_{11}$. The Sub-HART will necessarily be smaller and easier to interpret. The choice of how to restrict the feature space can be motivated by the specific problem setting. For example, consider fitting HAL to predict the recurrence of breast cancer based on demographic features and features related to the original tumors. If we are mainly interested in how the estimated function classifies older subjects, we could restrict Age ≥ 65. If we are interested in how the function differs between older and younger subjects, we could compare Sub-HARTs restricting Age < 20 and Age ≥ 65 respectively.

Algorithm 2 Bin Predictions

```
def Bin(Tree, Q):
intervals = {[0, 1/Q), [1/Q, 2/Q), ..., [(Q−1)/Q, 1]}
if Tree is TerminalNode then
    max = 1/Q · ceiling(Tree.pred / Q)
    min = max − 1/Q
    return TerminalNode([min, max))
else if All Tree.preds in i for i in intervals then
    return TerminalNode(i)
else
    node = Node(Tree.label)
    left-subtree = Bin(Tree.left-child, Q)
    right-subtree = Bin(Tree.right-child, Q)
    left-subtree.parent = node
    right-subtree.parent = node
    return(node)
end if
```

Algorithm 3 Aggregate Predictions

```
if Tree is TerminalNode then
    return TerminalNode(Tree.pred)
else if max(Tree.preds) − min(Tree.preds) ≤ K then
    interval = [min(Tree.preds), max(Tree.preds)]
    return TerminalNode(interval)
else
    node = Node(Tree.label)
    left-subtree = Agg(Tree.left-child, Q)
    right-subtree = Agg(Tree.right-child, Q)
    left-subtree.parent = node
    right-subtree.parent = node
    return(node)
end if
```

A second strategy is to smooth the predicted outcomes so that there are fewer unique predictions to display. We can then collapse regions of the tree that no longer show any heterogeneity. HART may display many adjacent terminal nodes with predictions that differ by small, clinically insignificant amounts. This level of granularity might unnecessarily add to the size of the tree. One method for smoothing predictions is to break the prediction space into a set number of intervals and only display the correct interval in each terminal node (Algorithm 2). For example, in the binary classification setting, we may bin the predictions into low ([0, .333)), medium ([0.333, 0.667)), and high ([0.667, 1.0)) probability of success. A second method is to choose some minimum amount of prediction heterogeneity required to introduce new splits (Algorithm 3). In the binary classification setting, we may enforce that splits only occur if they separate predicted probabilities having a difference higher than, say, 0.20. If a region of the tree has many terminal nodes and yet the predicted probabilities range only from .05 to 0.25, we can group that region into a single terminal node and display the prediction range. Even if one is still interested in viewing the predictions at the most granular level, a HART with smoothed predictions could be a convenient first step to identify interesting sub-regions to investigate further.

7 Algorithm complexity

Here we examine the theoretical time complexities of Algorithms 1, 2, and 3.

7.1 HART complexity

The exact time complexity of Algorithm 1 is difficult to specify as it depends on the proportion of candidate splits pruned after each node creation in the tree. That number is entirely dependent on the chosen splitting policy π and the fitted HAL model. Our analysis here is therefore limited to a range of worst-case and more optimistic scenarios.

We begin by considering the per-node computation in terms of the number of initial split candidates, which we will refer to as c. Given a split value, a node is created first by updating the set $\mathcal{R}$ to pinpoint the feature space region being considered. That involves comparing the split value to a set of values on the order of $O(n)$ within the same feature as the split value. Once the region is updated we shrink the candidate set which, involves looking at all candidates. This is a computation on the order of $O(c)$. We then take the remaining list of candidates, and compute a score for each based on the model fit information. In the case of π_{gini} that score is the gini impurity induced by splitting the data on that candidate. In the case of π_h that is the number of times the given candidate appears in the HAL basis expansion.

Table 2 UCI dataset characteristics. N is sample size, p is number of features

Name	Citation	N	p	$P_N(Y=1)$
Breast Cancer	Zwitter et al. [23]	285	9	0.298
Cardio	Ayres-de Campos et al. [24]	2126	21	0.139
Drugs	Fehrman et al. [25]	1885	12	0.186
Wine	Aeberhard et al. [26]	6497	12	0.197

Since we must examine each candidate once, this computation is on the order of $O(c)$. Once the best candidate is identified, we create a new node in the tree which is done in constant time, $O(1)$. We perform this set of operations for each non-terminal node in tree. In the absolute worse case in which only the chosen split is pruned at each stage, we have 2^{c-1} non-terminal nodes. In the case of a balanced, binary tree in which half of the candidates are pruned after each split, there are $2^{\log_2(c)-1} = c$ non-terminal nodes. Depending on the splitting policy π and the fitted model, there may be cases in which more or fewer than half of the candidates are pruned after each split. Thus, we estimate that the overall time complexity of Algorithm 1 is between the worst case of $O(nc^2 \cdot 2^{c-1})$ and a more optimistic case of $O(nc^3)$.

7.2 Bin and aggregate predictions complexity

Next we discuss the complexity of Algorithms 3 and 2. In both of these algorithms, a constant amount of work is done for each non-terminal node examined. In the worst case scenario, all non-terminal nodes in the tree are examined, although in practice it will be fewer if there is any binning or aggregation to be done. Therefore, the worst case time complexity is $O(t)$ where t is the total number of non-terminal nodes.

8 Data analysis

We examined the predictive performance of HAL, CART, Random Forest (Breiman, 2001), and XGBoost (Chen and Guestrin, 2016) using four publicly available data sets from the UCI Machine Learning Repository (Table 2) all of which have binary outcomes. Next, we focus on one data set and compare the decision tree produced by CART to those produced by HART using the heuristic policy shown in Eq. (2).

8.1 Performance

For each dataset in Table 2, HAL, CART, Random Forest, and XGBoost were evaluated by calculating several 10-fold cross-validated performance metrics. For each of CART, Random Forest, and XGBoost, models were built under a

Table 3 10-fold cross-validated metrics are reported for Breast Cancer, Cardio, and Drugs

Data	Learner	AUC	Accuracy	Precision	Recall	Time
Breast cancer	HAL	0.710	0.751	0.709	0.307	13.415 sec
	CART	0.669	0.702	0.500	0.366	0.297 sec
	XGboost	0.712	0.737	0.593	0.376	13.43 min
	Random forest	0.696	0.722	0.588	0.235	32.882 sec
Cardio	HAL	0.973	0.953	0.886	0.763	1.73 min
	CART	0.933	0.935	0.798	0.712	0.611 sec
	XGboost	0.981	0.961	0.898	0.810	46.89 min
	Random forest	0.979	0.950	0.879	0.739	3.594 min
Drugs	HAL	0.748	0.814	0.604	0.057	28.58 min
	CART	0.662	0.762	0.306	0.220	0.635 sec
	XGboost	0.746	0.814	0.486	0.051	39.453 min
	Random forest	0.741	0.816	0.615	0.023	3.472 mins
Wine	HAL	0.973	0.953	0.886	0.763	1.76 h
	CART	0.813	0.817	0.537	0.482	0.380 sec
	XGboost	0.904	0.874	0.741	0.554	30.380 min
	Random forest	0.921	0.888	0.862	0.510	3.563 min

3-fold cross validated metrics are reported for Wine. For CART, Random Forest, and XGBoost, we carried out grid searches over 10 tuning parameter settings. Results correspond to models with the highest performing tuning parameters. Computations were carried out on a High Performance Computing cluster

Table 4 Breast Cancer dataset feature names and descriptions

Feature	Description
Age	Discretized subject age in years
Early meno	Whether the subject reach menopause early
Pre-meno	Whether the subject is pre-menopausal
Tumor size	Size of breast cancer tumor in mm
Aux-nodes	# auxiliary lymph nodes containing metastatic cancer
In node cap	Whether metastatic tumors are encased in lymph node capsule
Deg-malig	Histological degree of the tumor malignancy (range 1-3)
Breast	Which breast cancer resides in
Breast quadrant	Quadrant of breast cancer resides in
Rad therapy	Whether the subject received radiation therapy

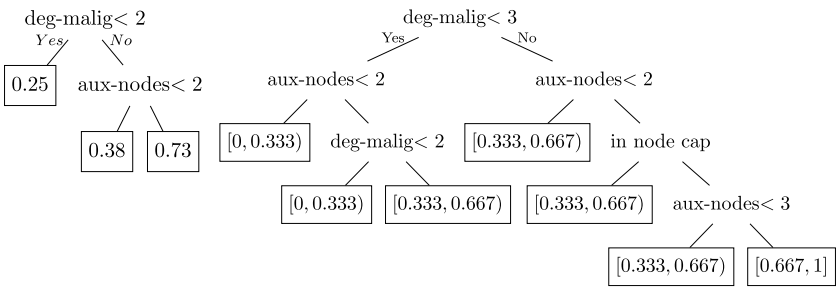

Fig. 7 Left: CART fit to Breast Cancer dataset. Right: HART built from HAL fit to Breast Cancer dataset. Feature space is restricted to subjects aged 20–29 having tumors greater than 5 mm in diameter. Algorithm 2 was applied with $Q = 3$

grid of 10 possible tuning parameter settings. We only report the results corresponding to models built with the parameter settings that resulted in the highest CV-AUCs for each learner. We found that HAL, Random Forest, and XGBoost typically provided large improvement over CART in all metrics (Table 3). Moreover, we found that the performance of HAL is comparable to Random Forest and XGBoost, which are considered state-of-the-art. For a more extensive examination of HAL's performance, see Benkeser and van der Laan (2016) [15].

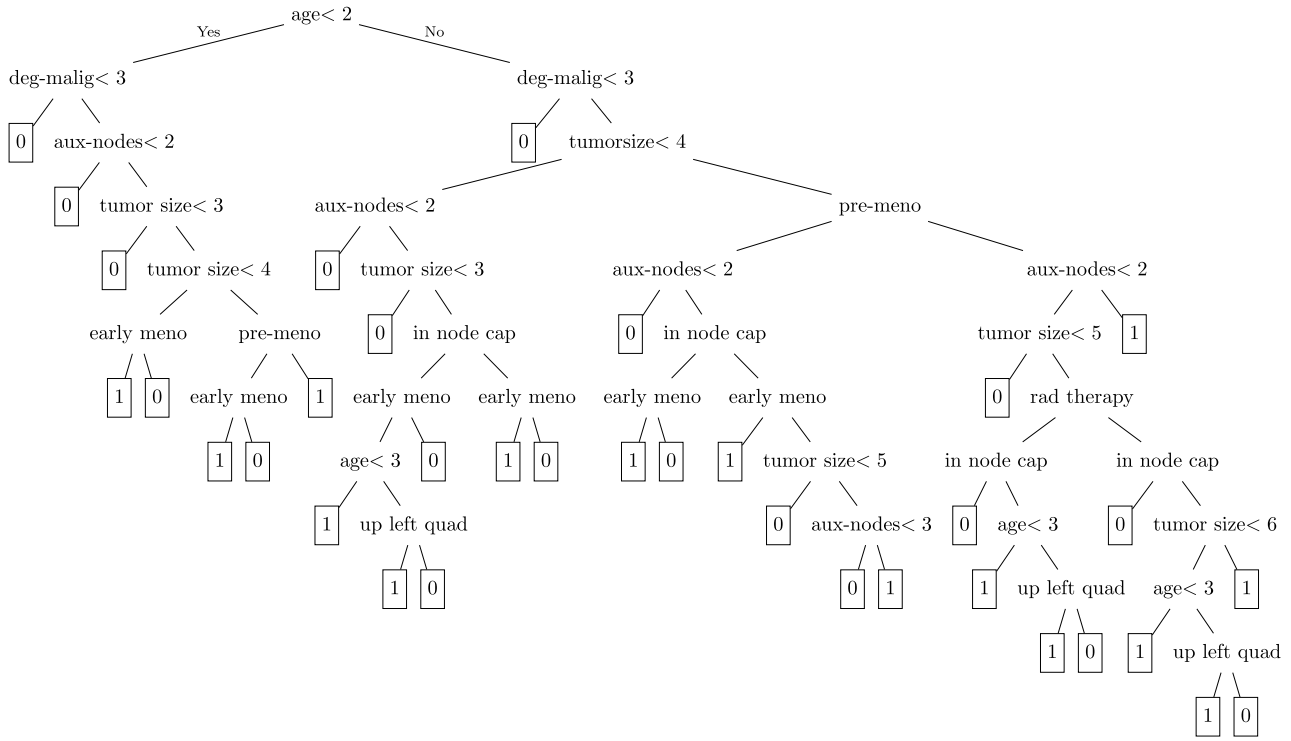

Fig. 8 HART representing HAL fit to Breast Cancer data. Algorithm 2 has been applied with $Q = 2$ bins, equivalent to thresholding the predicted probability of recurrence at 0.5

8.2 Trees

Here we focus on the Breast Cancer dataset to demonstrate the interpretability of HART. We aim to predict the recurrence of breast cancer with nine features, descriptions for which can be found in Table 4. Feature names and descriptions for the other three datasets can be found in the Supplement. The left panel of Fig. 7 shows the tree built using CART with tuning parameters selected using 10-fold cross-validation. Here, the fitted CART model implies that, of the 9 available features, only the degree of malignancy and the number of auxiliary lymph nodes containing metastatic breast cancer are needed to make a prediction about risk for breast cancer recurrence. However, the HART built using the same data shows a much more complex function and has much higher performance. The right panel of Fig. 7 shows a Sub-HART visualized for subjects aged $20 - 29$ having tumors greater than 5 mm in diameter. We have binned the predicted probabilities into three intervals: [0, 0.333), [0.333, .667), and [0.667, 1]. This model implies a higher degree of heterogeneity in the predicted probability that relies on more features and values. We could investigate any of the terminal regions in Fig. 7 further without binning the predictions. Alternatively, we could restrict to subjects older than 29 to see how the function changes with age.

If we are interested in using HART to make binary decisions based on a probability threshold, we can simplify the overall tree and visualize it without restricting the feature space. Figure 8 visualizes the full HART with 1 displayed in regions where predicted probabilities are above 0.5 and 0 displayed otherwise. Again, we see that increased performance is associated with a much more complex function. However, we can gain insights into HART's structure that could motivate different visualizations or even further study. Consider the role of menopause status in the prediction. Figure 8 suggests that having reached menopause before age 40 is associated with lower risk for breast cancer recurrence versus having reached it after 40 or being pre-menopause. This association is consistent with breast cancer research [27].

8.3 Complexity

We examined the time it took to apply Algorithms 1, 2, and 3 to both the Breast Cancer and Drugs datasets. After fitting HAL to the Breast Cancer data, there were 15 split candidates. The full HART constructed from the model fit was constructed in 0.752 s. Binning and Aggregating the predictions in the resulting tree took 0.015 s each. Applying the algorithms to the Cardio dataset took significantly longer. After fitting HAL to the Cardio data, there were 104 candidate splits. Constructing the full HART took 20.210 min. Binning and Aggregating the predictions in the resulting tree took 1.103 and 1.215 min respectively. All computations were carried out on an Apple Macbook Air containing an ARM processor with 8GB of RAM.

9 Discussion

In this paper we have presented a tool for post-hoc interpretation of the Highly Adaptive Lasso. HAL has the potential to learn more complex functions than CART without overfitting the data. HART, via Algorithms 1, 2, and 3, provides methodology for understanding these complex functions. An additional advantage of HART is that it allows one to tailor visualizations to the needs of the problem via the splitting policy π and Algorithms 2. In general, we recommend that HART be used less as a static tool for visualizing simple decision making processes and more as a dynamic way to visualize and understand a complex decision making process. In the future, it would be of interest to apply HART to problems that require transparent decision making such as medical treatment assignment and fair social policy design.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12065-023-00836-0.

Acknowledgements All figures were created using the Tikz [28] and Forest [29] packages in Latex.

Author Contributions SN and DB both conceptualized the research, developed the methodology, conducted the analyses, and wrote and edited this work.

Funding The research leading to these results received funding from the National Science Foundation under Grant Agreement No 2015540.

Data availibility The datasets analysed during the current study are all available in the public UCI Machine Learning Repository, (https://archive.ics.uci.edu/ml/index.php). Individual links are provided below. Breast Cancer: https://archive.ics.uci.edu/ml/datasets/breast+cancer Cardio: https://archive.ics.uci.edu/ml/datasets/cardiotocography Drugs: https://archive.ics.uci.edu/ml/datasets/Drug+consumption+ Wine: https://archive.ics.uci.edu/ml/datasets/wine+quality.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Consent to participate Not applicable. The research leading to these results did not involve any live subjects.

Consent for publication All authors and relevant institutions have given consent for this work to be published.

Code availability The code used to produce these results is available in the Supplementary materials.

References

1. Pirracchio R, Petersen ML, Carone M, Rigon MR, Chevret S, van der Laan MJ (2015) Mortality prediction in intensive care units with the super icu learner algorithm (sicula): a population-based study. Lancet Respir Med 3(1):42–52. https://doi.org/10.1016/S2213-2600(14)70239-5
2. Churpek MM, Yuen TC, Winslow C, Meltzer DO, Kattan MW, Edelson DP (2016) Multicenter comparison of machine learning methods and conventional regression for predicting clinical deterioration on the wards. Crit Care Med 44(2):368. https://doi.org/10.1097/CCM.0000000000001571
3. Acion L, Kelmansky D, van der Laan M, Sahker E, Jones D, Arndt S (2017) Use of a machine learning framework to predict substance use disorder treatment success. PLOS One 12(4):0175383. https://doi.org/10.1371/journal.pone.0175383
4. Rosellini AJ, Dussaillant F, Zubizarreta JR, Kessler RC, Rose S (2018) Predicting posttraumatic stress disorder following a natural disaster. J Psychiatr Res 96:15–22. https://doi.org/10.1016/j.jpsychires.2017.09.010
5. Bajari P, Nekipelov D, Ryan SP, Yang M (2015) Machine learning methods for demand estimation. Am Econ Rev 105(5):481–85
6. Amat C, Michalski T, Stoltz G (2018) Fundamentals and exchange rate forecastability with simple machine learning methods. J Int Money Finance 88:1–24
7. Zeineddine H, Braendle U, Farah A (2021) Enhancing prediction of student success: automated machine learning approach. Comput Electr Eng 89:106903
8. Huang X-L, Ma X, Hu F (2018) Machine learning and intelligent communications. Mob Netw Appl 23(1):68–70
9. Kusner MJ, Loftus JR, Russell C, Silva R (2017) Counterfactual fairness. ArXiv e-prints (arXiv:1703.06856 [stat.ML]. https://doi.org/10.48550/arXiv.1703.06856
10. Podgorelec V, Kokol P, Stiglic B, Rozman I (2002) Decision trees: an overview and their use in medicine. J Med Syst 26(5):445–463. https://doi.org/10.1023/a:1016409317640
11. Chern C-C, Chen Y-J, Hsiao B (2019) Decision tree-based classifier in providing telehealth service. BMC Med Inform Decis Mak 19(1):1–15. https://doi.org/10.1186/s12911-019-0825-9
12. Venkatasubramaniam A, Wolfson J, Mitchell N, Barnes T, JaKa M, French S (2017) Decision trees in epidemiological research. Emerg Themes Epidemiol 14(1):1–12. https://doi.org/10.1186/s12982-017-0064-4
13. Zhang H, Legro RS, Zhang J, Zhang L, Chen X, Huang H, Casson PR, Schlaff WD, Diamond MP, Krawetz SA (2010) Decision trees for identifying predictors of treatment effectiveness in clinical trials and its application to ovulation in a study of women with

polycystic ovary syndrome. Human Reprod 25(10):2612–2621. https://doi.org/10.1093/humrep/deq210
14. Friedman J, Hastie T, Tibshirani R (2001) The Elements of Statistical Learning vol. 1. Springer, ???
15. Benkeser D, Van Der Laan M (2016) The highly adaptive lasso estimator. In: 2016 IEEE international conference on data science and advanced analytics (DSAA), IEEE. pp 689–696 https://doi.org/10.1109/DSAA.2016.93
16. van der Laan M (2017) A generally efficient targeted minimum loss based estimator based on the highly adaptive lasso. Int J Biostat. https://doi.org/10.1515/ijb-2015-0097
17. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J Royal Stat Soc Series B 58(1):267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x
18. Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, Boca Raton
19. Bollig B, Wegener I (1996) Improving the variable ordering of obdds is np-complete. IEEE Trans Comput 45(9):993–1002. https://doi.org/10.1109/12.537122
20. Salzberg SL (1994) C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. Kluwer Academic Publishers
21. Kass GV (1980) An exploratory technique for investigating large quantities of categorical data. J Royal Stat Soc Series C (Appl Stat) 29(2):119–127. https://doi.org/10.2307/2986296
22. Oliver JJ, Dowe DL, Wallace C (1992) Inferring decision graphs using the minimum message length principle. In: Proceedings of the 5th Australian joint conference on artificial intelligence, World Scientific. pp 361–367
23. Zwitter M, Matjaz (1988) Soklic: breast cancer. UCI Machine Learning Repository
24. Ayres-de-Campos D, Bernardes J, Garrido A, Marques-de-Sa J, Pereira-Leite L (2000) Sisporto 2.0: a program for automated analysis of cardiotocograms. J Matern-Fetal Med 9(5):311–318. https://doi.org/10.1002/1520-6661
25. Fehrman E, Muhammad AK, Mirkes EM, Egan V, Gorban AN (2017) The five factor model of personality and evaluation of drug consumption risk. In: Data Science, pp. 231–242. Springer, ??? https://doi.org/10.1007/978-3-319-55723-6_18
26. Aeberhard S, Coomans D, De Vel O (1992) Comparison of classifiers in high dimensional settings. Dept Math Statist, James Cook Univ., North Queensland, Australia, Tech. Rep **92**(02) https://doi.org/10.1016/0031-3203(94)90145-7
27. Lao C, Elwood M, Kuper-Hommel M, Campbell I, Lawrenson R (2021) Impact of menopausal status on risk of metastatic recurrence of breast cancer. Menopause 28(10):1085–1092. https://doi.org/10.1097/GME.0000000000001817
28. Tantau T The TikZ and PGF Packages. https://tikz.dev/
29. Zivanovic S Forest. https://doi.org/10.5281/zenodo.1234. https://github.com/sasozivanovic/forest