Understanding Interventional TreeSHAP : How and Why it Works

Gabriel Laberge, Yann Batiste Pequignot

Understanding Interventional TreeSHAP : How and Why it Works.

Gabriel Laberge gabriel.laberge@polymtl.ca
Génie Informatique et Génie Logiciel
Polytechnique Montréal

Yann Pequignot yann.pequignot@iid.ulaval.ca
Institut Intelligence et Données
Université de Laval à Québec

Abstract

Shapley values are ubiquitous in interpretable Machine Learning due to their strong theoretical background and efficient implementation in the SHAP library. Computing these values previously induced an exponential cost with respect to the number of input features of an opaque model. Now, with efficient implementations such as Interventional TreeSHAP, this exponential burden is alleviated assuming one is explaining ensembles of decision trees. Although Interventional TreeSHAP has risen in popularity, it still lacks a formal proof of how/why it works. We provide such proof with the aim of not only increasing the transparency of the algorithm but also to encourage further development of these ideas. Notably, our proof for Interventional TreeSHAP is easily adapted to Shapley-Taylor indices and one-hot-encoded features.

Keywords: TreeSHAP, Decision Trees, Shapley Values, Attributions, Explainability

1. Introduction

Ever since their introduction, Random Forests (Breiman, 2001) and Gradient-Boosted-Trees (Friedman, 2001) have retained state-of-the-art performance for Machine Learning (ML) tasks on structured data (Grinsztajn et al., 2022). Still, by aggregating ensembles of decision trees rather than considering a single one, both models do away with the inherent transparency of decision trees in favor of more opaque decision-making mechanisms. Despite having high performance, the black box nature of these models is a considerable hurdle to their widespread applications as humans are not inclined to use tools that they cannot understand/explain (Arrieta et al., 2020).

To this end, research on the subject of explaining opaque models has been flourishing in the past few years. Notably, a method called SHAP (Lundberg and Lee, 2017) has recently been invented to provide explanations of any model decisions in the form of feature attributions, where a score (positive/negative or null) is assigned to each input feature and is meant to convey how much said feature was used in a specific model decision. These scores are based on the well-known Shapley values from coalitional game theory (Shapley, 1953). However, computing these Shapley values for arbitrary models induces an exponential cost in terms of the number of input features, which inhibits their application to most ML tasks.

The exponential burden of Shapley values has lately been alleviated for tree ensemble models (*e.g.* Random Forests and Gradient-Boosted-Trees) via two efficient algorithms

called Interventional TreeSHAP and Conditional TreeSHAP (Lundberg et al., 2020). The two algorithms differ in how their game theory formulation handles inputs with missing features. The invention of these algorithms has played a major role in making the SHAP Python library the most popular framework for shedding light on model decisions (Holzinger et al., 2022). Yet, the original paper is limited to the presentation of the algorithms, without a mathematical proof to justify their correctness. Some effort has subsequently been made to explain the main intuitions on how Interventional TreeSHAP works, as witnessed for example in the blog of Chen et al. (2020) which contains nice interactive visualizations. Still, to the best of the author's knowledge, a detailed proof that Interventional TreeSHAP works has not yet been written. This is the purpose of this paper and we believe it is important for multiple reasons.

- **Educational:** A correctness proof for an algorithm can be used as content for a math-focused class on interpretable ML.
- **Transparency:** It brings to light the assumptions made by the algorithm and the theoretical quantities it is computing. It alleviates the possibility of practitioners treating TreeSHAP as "a black box to explain a black box".
- **Transferability:** The proof for an algorithm allows one to derive proofs for similar algorithms.

The main objective of this paper is to provide the first complete proof of how/why TreeSHAP works. We will focus on the "Interventional" setting of TreeSHAP, where features are perturbed without consideration to the joint data distribution. The reason is that this algorithm is used by default any time practitioners provide a reference dataset to TreeSHAP[1]. Our contributions are the following:

1. We provide the first detailed proof of how/why Interventional TreeSHAP works.
2. The proof is shown to be effortlessly transferable to other Game Theory indices like the Shapley-Taylor indices (Sundararajan et al., 2020). Moreover, the proof is extended to features that are one-hot encoded, something that is not currently supported in the SHAP library.
3. We provide C++ implementations for Taylor-TreeSHAP, and Partition-TreeSHAP wrapped in Python[2]. Crucially, this C++ implementation uses the same notation as in the paper to serve as complementary resources for the interested reader.

The rest of the paper is structured as follows. We begin by introducing Shapley values in **Section 2**, and Decision Trees in **Section 3**. Then, **Section 4** presents our proof of correctness for the TreeSHAP algorithm. Next, our theoretical results are leveraged to effortlessly generalize TreeSHAP to the Shapley-Taylor index in **Section 5**, and to one-hot encoded features in **Section 6**. Finally, **Section 7** concludes the paper.

1. https://github.com/slundberg/shap/blob/45b85c1837283fdaeed7440ec6365a886af4a333/shap/explainers/_tree.py#L69
2. https://github.com/gablabc/Understand_TreeSHAP

2. Feature Attribution via Shapley Values

2.1 Shapley Values

Coalitional Game Theory studies situations where d players collaborate toward a common outcome. Formally, letting $[d] := \{0, 1, \ldots, d-1\}$ be the set of all d players, this theory is concerned with games $\nu : 2^{[d]} \to \mathbb{R}$, where $2^{[d]}$ is the set of all subsets of $[d]$, which describes the collective payoff that any set of players can gain by forming a coalition. In this context, the challenge is to assign a credit (score) $\phi_i(\nu) \in \mathbb{R}$ to each player $i \in [d]$ based on their contribution toward the total value $\nu([d]) - \nu(\emptyset)$ (the collective gain when all players join, taking away the gain when no one joins). Namely, such scores should satisfy:

$$\sum_{i=1}^{d} \phi_i(\nu) = \nu([d]) - \nu(\emptyset). \tag{1}$$

The intuition behind this Equation is to think of the outcomes $\nu([d])$ and $\nu(\emptyset)$ as the profit of a company involving all employees and no employee. In that case, the score $\phi_i(\nu)$ can be seen as the salary of employee i based on their productivity. There exist infinitely many score functions that respect Equation 1, hence it is necessary to define other desirable properties that a score function should possess: Dummy, Symmetry, and Linearity.

Dummy In cooperative games, a player i is called a *dummy* if $\forall S \subseteq [d] \setminus \{i\}\ \ \nu(S \cup \{i\}) = \nu(S)$. Dummy players basically never contribute to the game. A desirable property of a score is that dummy players should not be given any credit, *i.e.* employees that do not work do not make a salary,

$$\Big[\forall S \subseteq [d] \setminus \{i\}\ \ \nu(S \cup \{i\}) = \nu(S)\Big] \Rightarrow \phi_i(\nu) = 0. \tag{2}$$

Symmetry Another property is symmetry which states that players with equivalent roles in the game should have the same score *i.e.* employees with the same productivity should have the same salary

$$\Big[\forall S \subseteq [d] \setminus \{i, j\} \quad \nu(S \cup \{i\}) = \nu(S \cup \{j\})\Big] \Rightarrow \phi_i(\nu) = \phi_j(\nu). \tag{3}$$

Linearity The last desirable property is that scores are linear w.r.t games, *i.e.* if we let $\boldsymbol{\phi}(\nu) := [\phi_1(\nu), \phi_2(\nu), \ldots, \phi_d(\nu)]^T$, then for all games $\nu, \mu : 2^{[d]} \to \mathbb{R}$ and all $\alpha \in \mathbb{R}$, we want

$$\boldsymbol{\phi}(\nu + \mu) = \boldsymbol{\phi}(\nu) + \boldsymbol{\phi}(\mu). \tag{4}$$

$$\boldsymbol{\phi}(\alpha\mu) = \alpha\boldsymbol{\phi}(\mu). \tag{5}$$

The reasoning behind this property is a bit more involved than the previous two. For Equation 4, imagine that the two games ν and μ represent two different companies and that employee i works at both. In that case, the salary of employee i from company ν should not be affected by their performance in the company μ and vice-versa. Importantly, the total salary of employee i ideally should be the sum of the salaries at both companies. For Equation 5, we imagine that the company is subject to a law-suit in the end of a

quarter which results in a sudden reduction in profits by a factor of α. Then, given that each employee had fixed productivity during this quarter, it is only fair that all their salaries should also diminish by the same factor α.

In his seminal work, Lloyd Shapley has proven the existence of a **unique** score function that respects Equations 1-5: the so-called Shapley values.

Definition 1 (Shapley values (Shapley, 1953)) *Given a set $[d] := \{1, 2, \ldots, d\}$ of players and a cooperative game $\nu : 2^{[d]} \to \mathbb{R}$, the Shapley values are defined as*

$$\phi_i(\nu) = \sum_{S \subseteq [d] \setminus \{i\}} W(|S|, d)\big(\nu(S \cup \{i\}) - \nu(S)\big) \qquad i \in [d], \tag{6}$$

where

$$W(k, d) := \frac{k!(d-k-1)!}{d!} \tag{7}$$

is the proportion of all $d!$ orderings of $[d]$ where a given subset S of size k appears first, directly followed by a distinguished element $i \notin S$.

Intuitively, the credit $\phi_i(\nu)$ is attributed to each player $i \in [d]$ based on the average contribution of adding them to coalitions S that excludes them. Notice that the number of terms in **Definition 1** is exponential in the number of players.

We record here a simple property of Shapley values with respect to dummy players that will be essential in our derivations.

Lemma 2 (Dummy Reduction) *Let $D \subseteq [d]$ be the set of all dummies, and $D^C = [d] \setminus D$ be the set of non-dummy players, then*

$$\phi_i(\nu) = \begin{cases} 0 & \text{if } i \in D \\ \sum_{S \subseteq D^C \setminus \{i\}} W(|S|, |D^C|)\big(\nu(S \cup \{i\}) - \nu(S)\big) & \text{otherwise.} \end{cases} \tag{8}$$

Simply put, dummy players are given no credit, and the credit of all remaining players follows the basic Shapley definition while assuming that the set of players is D^C instead of $[d]$.

Proof Let j be a dummy but not i, we have

$$\begin{aligned} \phi_i(\nu) &= \sum_{S \subseteq [d] \setminus \{i,j\}} W(|S|, d)\big(\nu(S \cup \{i\}) - \nu(S)\big) + W(|S|+1, d)\big(\nu(S \cup \{i,j\}) - \nu(S \cup \{j\})\big) \\ &= \sum_{S \subseteq [d] \setminus \{i,j\}} \big(W(|S|, d) + W(|S|+1, d)\big)\big(\nu(S \cup \{i\}) - \nu(S)\big) \\ &= \sum_{S \subseteq [d] \setminus \{i,j\}} W(|S|, d-1)\big(\nu(S \cup \{i\}) - \nu(S)\big). \end{aligned}$$

Repeat this process for all dummies. ■

This property is key as it implies that Shapley values of non-dummy players are unaffected by the inclusion of an arbitrarily large number of new dummy players. It also means that, once all dummy players are identified, they can be ignored and the Shapley value computations will only involve the set D^C non-dummy players.

2.2 Feature Attribution as a Game

For a long time, the focus of Machine Learning (ML) has been on improving generalization performance, regardless of the cost of model interpretability. For instance, because of their state-of-the-art performance, tree ensembles are typically used instead of individual decision trees which are inherently more interpretable. However, the ML community has reached a point where the lack of transparency of the best-performing models inhibits their widespread acceptance and deployment (Arrieta et al., 2020). To address this important challenge, the research community has been working for the past few years on techniques that shed light on the decision-making of opaque models. A popular subset of such techniques provide insight on model decisions in the form of feature attribution *i.e.* given an opaque model h and an input $\boldsymbol{x} \in \mathbb{R}^d$, a score $\phi_i(h, \boldsymbol{x}) \in \mathbb{R}$ is provided to each feature $i \in [d]$ in order to convey how much feature i was used in the decision $h(\boldsymbol{x})$. Given the lack of ground truth for the feature attribution for an opaque model, the community has focused on exploiting the existence and uniqueness of Shapley values to the problem of explaining ML models (Sundararajan and Najmi, 2020).

Following the BaselineShap formulation of Sundararajan and Najmi (2020), when explaining the model prediction $h(\boldsymbol{x})$ at some input $\boldsymbol{x}$ of interest, we can compare said prediction to the prediction $h(\boldsymbol{z})$ at a baseline value of the input $\boldsymbol{z}$. In this context, feature attributions can be formulated as feature contributions towards the gap $h(\boldsymbol{x}) - h(\boldsymbol{z})$ viewed as the total value of a carefully designed coalitional game. Defining such a game requires specifying the collective payoff for every subset of features. In BaselineShap, this is achieved by defining the collective payoff for a given set S of features to be the prediction of the model at the input whose value for features from S are given by the point of interest $\boldsymbol{x}$, while other feature values are taken from the baseline point $\boldsymbol{z}$. Formally, given an input of interest $\boldsymbol{x}$, a subset of features $S \subseteq [d]$ that are considered active, and the baseline $\boldsymbol{z}$, the replace function $\boldsymbol{r}_S^{\boldsymbol{z}} : \mathbb{R}^d \to \mathbb{R}^d$, defined as

$$r_S^{\boldsymbol{z}}(\boldsymbol{x})_i = \begin{cases} x_i & \text{if } i \in S \\ z_i & \text{otherwise,} \end{cases} \tag{9}$$

is used to simulate the action of activating features in S (or equivalently shutting down features not in S). The resulting coalitional game for ML interpretability is the following.

Definition 3 (The Baseline Interventional Game) *Comparing the model predictions at $\boldsymbol{x}$ and $\boldsymbol{z}$ can be done by computing the Shapley values of the following game:*

$$\nu_{h,\boldsymbol{x},\boldsymbol{z}}(S) := h(\, \boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})\,). \tag{10}$$

This formulation is called Interventional because we intervene on feature i by replacing its baseline value z_i with the value x_i irrespective of the value of other features. Hence we break correlations between the various features.

By Equation 1, the resulting Shapley values $\boldsymbol{\phi}(\nu_{h,\boldsymbol{x},\boldsymbol{z}}) \equiv \boldsymbol{\phi}(h, \boldsymbol{x}, \boldsymbol{z})$ will sum to the gap

$$\sum_{i=1}^{d} \phi_i(h, \boldsymbol{x}, \boldsymbol{z}) = h(\boldsymbol{x}) - h(\boldsymbol{z}). \tag{11}$$

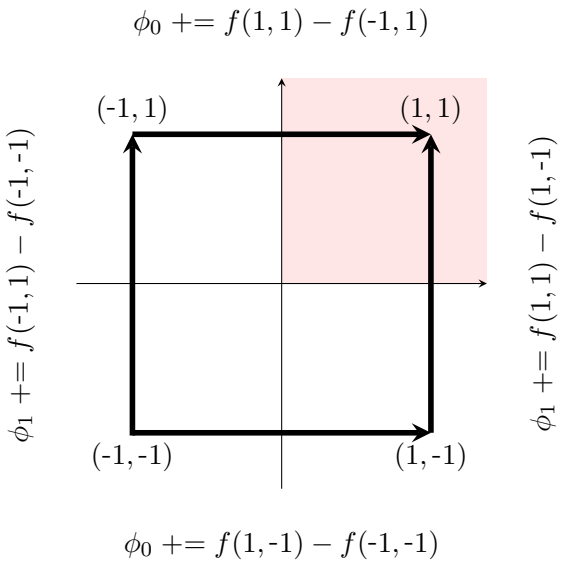


Figure 1: Example of Shapley values computation.

Therefore, Shapley values following this Interventional game formulation are advertised as a mean to answer contrastive questions such as : *Why does my model attribute a higher risk of stroke to person A compared to person B?*

We finish this Section with a toy example that illustrates how Interventional Shapley values are computed. We shall be explaining a two-dimensional `AND` function

$$h(\boldsymbol{x}) = \mathbb{1}(x_0 > 0)\mathbb{1}(x_1 > 0). \tag{12}$$

Our goal is to explain the discrepancy between $h(\boldsymbol{x}) = 1$ with $x = (1,1)^T$ and $h(\boldsymbol{z}) = 0$ with $\boldsymbol{z} = (\text{-}1,\text{-}1)^T$. Let us begin by computing $\phi_0(h, \boldsymbol{x}, \boldsymbol{z})$ by directly plugging in the definition

$$\begin{aligned}
\phi_0(h,\boldsymbol{x},\boldsymbol{z}) &= \sum_{S\subseteq\{1\}} \frac{|S|!(d-|S|-1)!}{d!}\big[\, h(\, \boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{0\}}(\boldsymbol{x})\,) - h(\, \boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x})\,)\,\big] \\
&= \frac{1}{2}\big[\, h(\, \boldsymbol{r}^{\boldsymbol{z}}_{\{0,1\}}(\boldsymbol{x})\,) - h(\, \boldsymbol{r}^{\boldsymbol{z}}_{\{1\}}(\boldsymbol{x})\,)\,\big] + \frac{1}{2}\big[\, h(\, \boldsymbol{r}^{\boldsymbol{z}}_{\{0\}}(\boldsymbol{x})\,) - h(\, \boldsymbol{r}^{\boldsymbol{z}}_{\emptyset}(\boldsymbol{x})\,)\,\big] \\
&= \frac{1}{2}\big(\, h(1,1) - h(\text{-}1,1)\,\big) + \frac{1}{2}\big(\, h(1,\text{-}1) - h(\text{-}1,\text{-}1)\,\big) \\
&= \frac{1}{2}\times 1 + \frac{1}{2}\times 0 = \frac{1}{2}.
\end{aligned} \tag{13}$$

Same thing for $\phi_1(h, \boldsymbol{x}, \boldsymbol{z})$. We see that $\phi_0(h, \boldsymbol{x}, \boldsymbol{z}) = \phi_2(h, \boldsymbol{x}, \boldsymbol{z})$, which is aligned with the symmetry property (cf. Equation 3), and that $\phi_0(h, \boldsymbol{x}, \boldsymbol{z}) + \phi_1(h, \boldsymbol{x}, \boldsymbol{z}) = 1 = h(\boldsymbol{x}) - h(\boldsymbol{z})$. Figure 1 shows the intuition behind this computation. We see there are two coordinate-parallel paths that go from $\boldsymbol{z}$ to $\boldsymbol{x}$. We average the contribution of changing the ith component across these two paths.

3. Tree-Based Models

3.1 Decision Tree

A directed graph $G = (N, E)$ is a set of nodes $n \in N$ and edges $e \in E \subset N \times N$. The node at the tail of e is $e_0 \in N$ while the node at the head of the edge is $e_1 \in N$. The node e_1 is called a child of e_0. A full binary tree is a rooted tree in which every node has exactly two children (a left one and a right one) or none. We view such a graph as a directed graph by making all its edges point away from the root. We denote a full binary tree by $T = (N, E)$ where $N = \{0, 1, \ldots, |N| - 1\}$ is the set of nodes and $E \subset N \times N$ is the set of directed edges and write $0 \in N$ for the root. An *internal node* is a node n which has two children $l < r$ and l is called the left child of n and r is called the right child of n. In this case, we say that (n, r) is a right edge and (n, l) is a left edge. A leaf is a node with no children and all leaves are stored in the set $L \subseteq N$.

Definition 4 (Binary Decision Tree) *A Binary Decision Tree on $\mathbb{R}^d$ is full binary tree $T = (N, E)$ in which every internal node n is labeled by a pair $(i_n, \gamma_n) \in [d] \times \mathbb{R}$ and every leaf $l \in L$ is labeled with a value v_l. For an internal node n, the label (i_n, γ_n) encodes the boolean function $R_n : \mathbb{R}^d \to \{0, 1\}$ given by $R_n(\boldsymbol{x}) := \mathbb{1}(x_{i_n} \leq \gamma_n)$.*

See Figure 2 for a simple example of Binary Decision Tree. Any Binary Decision Tree induces a function $h : \mathbb{R}^d \to \mathbb{R}$ defined as follows. For every $\boldsymbol{x} \in \mathbb{R}^d$, we start at the top of the tree (the root) and check the condition $R_0(\boldsymbol{x}) := \mathbb{1}(x_{i_0} \leq \gamma_0)$: if it is true we go down to the left child of the root, and if it is false we go down to the right child. This procedure is repeated until we reach a leaf node l and the model outputs $h(\boldsymbol{x}) = v_l$. For instance, in Figure 2, we see that for the input $\boldsymbol{x} = (3.4, 0.2, 2)^T$, we go from the root to the node 1 and end up at the leaf 4, so $h(\boldsymbol{x}) = v_4$. We note that the input "flowed through" the sequence of edges $(\,(0, 1), (1, 4)\,)$ which goes from root to leaf. We shall refer to such a sequence as a maximal path.

Definition 5 (Maximal Path) *A directed path P in a full binary tree $T = (N, E)$ is sequence $P = (e^{[k]})_{k=0}^{\ell-1}$ of edges in E such that the $e_1^{[k]} = e_0^{[k+1]}$ for all $k = 0, 1, \ldots, \ell - 2$. A maximal path is a directed path $P = (e^{[k]})_{k=0}^{\ell-1}$ that cannot be extended (and if it has a positive length then it starts at the root and ends at a leaf).*

Examples of maximal paths in this decision tree of Figure 2 include $P = (\,(0, 1), (1, 4)\,)$ and $P = (\,(0, 2), (2, 5)\,)$. Given the definition of maximal path, we now aim at describing the model h in a more formal manner. Since our intuition about h is that $\boldsymbol{x}$ flows downward on edges selected by the Boolean functions, our notation should only involve edges. Hence, for an edge $e = (e_0, e_1)$, we define i_e as the feature index of the internal node e_0 at its tail. In Figure 2, we have that $i_{(1,4)} = 2$ for example. Also, for each edge $e = (e_0, e_1)$ we define the Boolean function $R_e(\boldsymbol{x})$ by $R_{e_0}(\boldsymbol{x})$ if e is a left edge and $1 - R_{e_0}(\boldsymbol{x})$ otherwise. When $R_e(\boldsymbol{x}) = 1$ for some edge e, we shall say that $\boldsymbol{x}$ "flows though" the edge e, a terminology that will be used throughout this paper. Now, for each maximal path P ending at some leaf $l \in L$ we define $h_P : \mathbb{R}^d \to \mathbb{R}$ by

$$h_P(\boldsymbol{x}) = v_l \prod_{e \in P} R_e(\boldsymbol{x}). \tag{14}$$

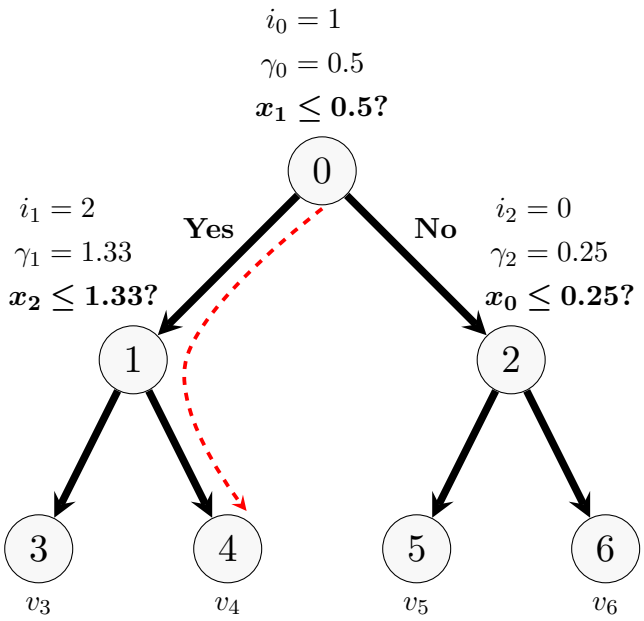


Figure 2: Basic example of Binary Decision Tree. In red we highlight the maximal path followed by the input $\boldsymbol{x} = (3.4, 0.2, 2)^T$.

This step function outputs zero unless the input flows through all the edges in the maximal path P. For example, in Figure 2, the function h_P associated with $P = (\,(0,1),(1,4)\,)$ is $h_P(\boldsymbol{x}) = v_4 \mathbb{1}(x_1 \leq 0.5)\mathbb{1}(x_2 > 1.33)$. Finally, given an input $\boldsymbol{x}$, there exists only one maximal path P such that $h_P(\boldsymbol{x}) \neq 0$. This is because any given $\boldsymbol{x}$ can only flow through one path P from root to leaf. Hence, we can interpret h as

$$h(\boldsymbol{x}) = \sum_P h_P(\boldsymbol{x}), \tag{15}$$

where the sum is taken over all maximal paths in the Decision Tree. Note that the decision tree function h of Figure 2 can be written as

$$\begin{aligned} h(\boldsymbol{x}) = \quad & v_3 \mathbb{1}(x_1 \leq 0.5)\mathbb{1}(x_2 \leq 1.33) + v_4 \mathbb{1}(x_1 \leq 0.5)\mathbb{1}(x_2 > 1.33) \\ & + v_5 \mathbb{1}(x_1 > 0.5)\mathbb{1}(x_0 \leq 0.25) + v_6 \mathbb{1}(x_1 > 0.5)\mathbb{1}(x_0 > 0.25). \end{aligned}$$

4. Interventional Tree SHAP

As previously highlighted, computing Shapley values (cf. **Definition 1**) requires summing over an exponential number of subsets $S \subseteq [d] \setminus \{i\}$. This exponential complexity previously prohibited the application of Shapley values to ML tasks which typically involve a large number of features. Nonetheless, by exploiting the structure of Decision Trees, Interventional TreeSHAP alleviates this exponential complexity in d at the cost of no longer being model-agnostic. We now describe step-by-step how and why TreeSHAP works.

In this section, we shall assume $\boldsymbol{x}$ and $\boldsymbol{z}$ are fixed and we are concerned with computing Shapley values for a forest $\mathcal{F}$ of decision trees, namely $\boldsymbol{\phi}(\mathcal{F}, \boldsymbol{x}, \boldsymbol{z})$. A first application of linearity (cf. Equations 4 & 5) shows that

$$\boldsymbol{\phi}(\mathcal{F}, \boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\phi}\left(\frac{1}{|\mathcal{F}|}\sum_{h\in\mathcal{F}} h, \boldsymbol{x}, \boldsymbol{z}\right) \tag{16}$$

$$= \frac{1}{|\mathcal{F}|}\sum_{h\in\mathcal{F}} \boldsymbol{\phi}(h, \boldsymbol{x}, \boldsymbol{z}), \tag{17}$$

Henceforth, we can thus fix a decision tree h and focus on how to compute $\boldsymbol{\phi}(h, \boldsymbol{x}, \boldsymbol{z})$.

4.1 Naive Tree Traversal

Since the function represented by the decision tree h can be written as a sum over all maximal paths (cf. Equation 15), a second application of linearity (cf. Equation 4) allows us to derive

$$\begin{aligned}\boldsymbol{\phi}(h, \boldsymbol{x}, \boldsymbol{z}) &= \boldsymbol{\phi}\left(\sum_P h_P, \boldsymbol{x}, \boldsymbol{z}\right) \\ &= \sum_P \boldsymbol{\phi}(h_P, \boldsymbol{x}, \boldsymbol{z}),\end{aligned} \tag{18}$$

Shapley values for a decision tree are therefore equal to the sum of Shapley values over all maximal paths in the tree. This means that the feature attributions can be computed via the following tree-traversal algorithm.

Algorithm 1 Tree SHAP Naive Pseudo-code

```
1: procedure RECURSE(n)
2:     if n ∈ L then
3:         φ += φ(h_P, x, z)          ▷ Add contribution of the maximal path
4:     else
5:         RECURSE(n_left child);
6:         RECURSE(n_right child);
7:     end if
8: end procedure
9: φ = 0;
10: RECURSE(0);
```

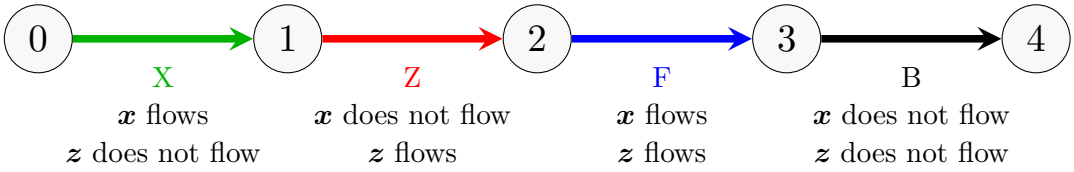


Figure 3: All types of edges in a maximal path $P = (\,(0,1),(1,2),(2,3),(3,4)\,)$ from root to leaf.

4.2 Compute the Shapley Value of a Maximal Path

We can now fix a maximal path P in the decision tree h and focus on the Shapley values $\boldsymbol{\phi}(h_P, \boldsymbol{x}, \boldsymbol{z})$. We first identify four types of edges X, Z, F, B which can occur in P.

Definition 6 (Edge Type) *We say that an edge $e \in E$ is of*

$$\begin{aligned} &\textit{Type } X \quad \textit{if} \quad R_e(\boldsymbol{x}) = 1 \textit{ and } R_e(\boldsymbol{z}) = 0 \\ &\textit{Type } Z \quad \textit{if} \quad R_e(\boldsymbol{x}) = 0 \textit{ and } R_e(\boldsymbol{z}) = 1 \\ &\textit{Type } F \quad \textit{if} \quad R_e(\boldsymbol{x}) = 1 \textit{ and } R_e(\boldsymbol{z}) = 1 \\ &\textit{Type } B \quad \textit{if} \quad R_e(\boldsymbol{x}) = 0 \textit{ and } R_e(\boldsymbol{z}) = 0. \end{aligned} \tag{19}$$

Here X stands for $\boldsymbol{x}$ flows, Z stands for $\boldsymbol{z}$ flows, F stands for both Flow, and B stands for both are Blocked.

See Figure 3 for an informal example of each of these types of edges for a fixed maximal path P. Moreover, Figure 4 presents the previous example of a simple decision tree where each edge is colored with respect to its type. From this point on in the document, we will systematically color edges w.r.t their type.

Lemma 7 *If P contains an edge of type B, then*

$$\forall\, S \subseteq [d] \quad h_P(\,\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})\,) = 0.$$

Proof Let e be an edge of type B in P. Then for every $S \subseteq [d]$, we have $R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = 0$, which implies that $h_P(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) \propto R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = 0$. ■

Corollary 8 *If P contains an edge of B , then $\boldsymbol{\phi}(h_P, \boldsymbol{x}, \boldsymbol{z}) = \mathbf{0}$.*

Proof It follows from **Lemma 7** and the linearity property of Shapley values (cf. Equation 5 with $\alpha = 0$). Indeed, by definition, linear functions map zero to zero so the Shapley values of the null game are null. ■

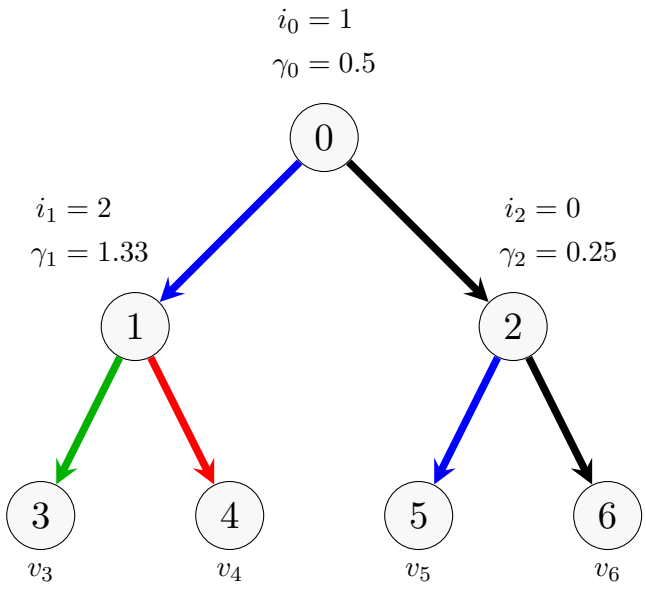


Figure 4: Example of types of edges in a Decision Tree. The input and reference inputs are $\boldsymbol{x} = (0, 0, 1)^T$ and $\boldsymbol{z} = (-2, -1, 2)^T$.

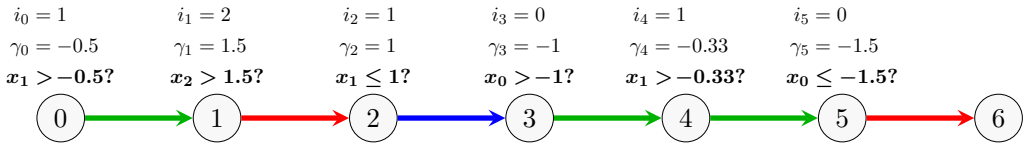


Figure 5: Example of sets $S_X = \{0, 1\}$ and $S_Z = \{0, 2\}$ for $\boldsymbol{x} = (0, 0, 1)^T$ and $\boldsymbol{z} = (-2, -1, 2)^T$.

Now, assuming P does not contain any type B edges, we have

$$\begin{aligned} h_P(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) &= v_l \prod_{e \in P} R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) \\ &= v_l \prod_{\substack{e \in P \\ e \text{ type X}}} R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) \prod_{\substack{e \in P \\ e \text{ type Z}}} R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) \underbrace{\prod_{\substack{e \in P \\ e \text{ type F}}} R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x}))}_{=1 \; \forall S} \\ &= v_l \prod_{\substack{e \in P \\ e \text{ type X}}} \mathbb{1}(i_e \in S) \prod_{\substack{e \in P \\ e \text{ type Z}}} \mathbb{1}(i_e \notin S). \end{aligned} \tag{20}$$

For an edge e, recall that i_e denotes the feature index at its tail e_0. The last line follows from the observation that, for an edge e of type X , the only times $R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = 1$ are when the i_eth component of $\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})$ is set to $\boldsymbol{x}$ and not $\boldsymbol{z}$ (*i.e.* $i_e \in S$). Alternatively, for an edge e of type Z , $R_e(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = 1$ if and only if the i_eth component of $\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})$ is set to $\boldsymbol{z}$ and not $\boldsymbol{x}$ (*i.e.* $i_e \notin S$).

We define the three following sets

$$\begin{aligned} S_X &:= \{i_e : e \in P, \text{ and } e \text{ is of type } X\}, \\ S_Z &:= \{i_e : e \in P, \text{ and } e \text{ is of type } Z\}. \\ S_{XZ} &:= S_X \cup S_Z. \end{aligned} \tag{21}$$

See Figure 5 for an example of sets S_X and S_Z on some maximal path. It is important to realize that the sets S_X, S_Z, and S_{XZ} depend on the current maximal path P. Still, we assume that P is fixed and hence we do not make the dependence explicit in the notation.

Lemma 9 *If $S_X \cap S_Z \neq \emptyset$, then*

$$\forall\, S \subseteq [d] \quad h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_S(\boldsymbol{x})\,) = 0. \tag{22}$$

Proof Let j be a feature common to both S_X and S_Z. Then, there must exists an edge e of X and an edge e' of type Z such that $j = i_e = i_{e'}$. From Equation 20, we have

$$h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_S(\boldsymbol{x})\,) \propto \mathbb{1}(j \in S)\mathbb{1}(j \notin S) = 0.$$

■

Corollary 10 *If $S_X \cap S_Z \neq \emptyset$, then $\boldsymbol{\phi}(h_P, \boldsymbol{x}, \boldsymbol{z}) = \mathbf{0}$.*

Proof By **Lemma 9** and Linearity of the Shapley values (cf. Equation 5). ■

We observe that **Corollary 8 & 10** provide sufficient conditions for the contributions at line 3 of Algorithm 1 to be null. We now provide a result that highlights which features are dummies of the Coalitional Game for the maximal path P.

Lemma 11 *If a feature does not belong to S_{XZ}, then it is a dummy feature for the Baseline Interventional Game $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}$ (cf. Definition 3).*

Proof If we assume there exists a type B edge in P, then by **Lemma 7**, the game is null and all players are dummies. In the non-trivial case where P does not contain a type B edge, we must let $i \notin S_{XZ}$ be an index, and $S \subseteq [d] \setminus \{i\}$ be an arbitrary subset of players that excludes i, and show that $h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})\,) = h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_S(\boldsymbol{x})\,)$. Since $i \notin S_{XZ}$, there are no edges e of type X or Z such that $i_e = i$. Moreover, since $i \neq i_e$ implies the equivalence $i_e \in S \iff i_e \in S \cup \{i\}$, we have

$$\begin{aligned} h_P(\boldsymbol{r}^{\boldsymbol{z}}_S(\boldsymbol{x})) &= v_l \prod_{\substack{e\in P \\ e \text{ type X}}} \mathbb{1}(i_e \in S) \prod_{e\in P e \text{ type Z}} \mathbb{1}(i_e \notin S) \\ &= v_l \prod_{\substack{e\in P \\ e \text{ type X}}} \mathbb{1}(i_e \in S \cup \{i\}) \prod_{\substack{e\in P \\ e \text{ type Z}}} \mathbb{1}(i_e \notin S \cup \{i\}) \\ &= h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})), \end{aligned}$$

which concludes the proof. ■

By **Lemma 2**, we note that computing the Shapley values $\boldsymbol{\phi}(h_P, \boldsymbol{x}, \boldsymbol{z})$ will only require evaluating the game $\nu_{h,\boldsymbol{x},\boldsymbol{z}}(S) := h_P(\, \boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})\,)$ for coalitions $S \subseteq S_{XZ}$. Additionally, we will assume w.l.o.g that the current maximal path contains no type B edges and that S_X and S_Z are disjoint. Indeed, failing to reach these requirements will simply cause the Shapley values to be zero.

Lemma 12 (Flow Blocking) *If P contains no type B edges and the sets S_X and S_Z are disjoint, then for any $S \subseteq S_{XZ}$ we have*

$$h_P(\, \boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})\,) = \begin{cases} v_l & \text{if } S = S_X \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

where $l \in L$ is the leaf node at the end of P.

Proof Since P does not contain a type B edge, we can use Equation 20, which, as a reminder, states that

$$h_P(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = v_l \prod_{\substack{e \in P \\ e \text{ type X}}} \mathbb{1}(i_e \in S) \prod_{\substack{e \in P \\ e \text{ type Z}}} \mathbb{1}(i_e \notin S). \tag{24}$$

We need to prove the following two statements.

1. $h_P(\boldsymbol{r}_{S_X}^{\boldsymbol{z}}(\boldsymbol{x})) = v_l$. On the one hand, for every edge e of type X in P, by definition $i_e \in S_X$. On the other hand, for every edge e of type Z in P, by definition $i_e \in S_Z$, so $i_e \notin S_X$ since we assume that $S_X \cap S_Z = \emptyset$. Therefore it follows that $h_P(\boldsymbol{r}_{S_X}^{\boldsymbol{z}}(\boldsymbol{x})) = v_l$ using Equation 24.

2. If $S \subseteq S_{XZ}$ and $S \neq S_X$, then $h_P(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = 0$. Assume that $S \subseteq S_{XZ}$ and $S \neq S_X$. Since, $S \neq S_X$, at least of the two following cases must occur, (a) there exists $j \in S_X \backslash S$ or (b) there exists $j \in S \setminus S_X$.

   (a) Let $j \in S_X \setminus S$. We can find $e \in P$ of type X such that $i_e = j$. We then have $\mathbb{1}(i_e \in S) = 0$.

   (b) Let $j \in S \setminus S_X$. Since $S \subseteq S_{XZ}$, it follows that $j \in S_Z$. We can find $e \in P$ of type Z such that $i_e = j$. Then as $j \in S$, we have $\mathbb{1}(i_e \notin S) = 0$.

   Given that either case (a) or case (b) must occur, using Equation 24 we see that $h_P(\boldsymbol{r}_S^{\boldsymbol{z}}(\boldsymbol{x})) = 0$, as desired.

■

We now arrive at the main Theorem which highlights how Interventional TreeSHAP computes Shapley values while avoiding the exponential cost w.r.t the number of input features. This Theorem is the culmination of all previous results : **Lemma 2**, **Corollary 8 & 10** and **Lemma 11 & 12**.

Theorem 13 (Complexity Reduction) *If P contains no type B edges and the sets S_X and S_Z are disjoint, then all features that are not in S_{XZ} are dummies and we get*

$$\phi_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = \sum_{S \subseteq S_{XZ} \setminus \{i\}} W(|S|, |S_{XZ}|)\big(h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x}))\big) \quad i \in S_{XZ}. \tag{25}$$

The exponential cost $\mathcal{O}(2^{|S_{XZ}|})$ of computing these terms reduces to $\mathcal{O}(1)$ following

$$i \notin S_{XZ} \Rightarrow \phi_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = 0 \tag{26}$$
$$i \in S_X \Rightarrow \phi_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = W(|S_X| - 1, |S_{XZ}|)v_l \tag{27}$$
$$i \in S_Z \Rightarrow \phi_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = -W(|S_X|, |S_{XZ}|)v_l, \tag{28}$$

given that the coefficients W where computed and stored in advance.

Proof Given **Lemma 11**, we know that all features not in S_{XZ} are dummies. Hence applying **Lemma 2** yields Equation 25 and 26. We now prove 27 and 28 separately.

Suppose $i \in S_X$, then according to **Lemma 12** the only coalition $S \subseteq S_{XZ}\setminus\{i\}$ for which $h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x}))$ is non-null is when $S = S_X \setminus \{i\}$. Indeed, when $S = S_X \setminus \{i\}$

$$\begin{aligned} h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x})) &= h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X\setminus\{i\}}(\boldsymbol{x})) \\ &= v_l - 0 = v_l. \end{aligned}$$

So, we get a Shapley value $W(|S|, |S_{XZ}|)v_l = W(|S_X \setminus \{i\}|, |S_{XZ}|)v_l = W(|S_X|-1, |S_{XZ}|)v_l$.

Now suppose $i \in S_Z$, according to **Lemma 12** the only coalition $S \subseteq S_{XZ} \setminus \{i\}$ for which $h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x}))$ is non-null is when $S = S_X$ Indeed, when $S = S_X$

$$\begin{aligned} h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x})) &= h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X\cup\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X}(\boldsymbol{x})) \\ &= 0 - v_l = -v_l. \end{aligned}$$

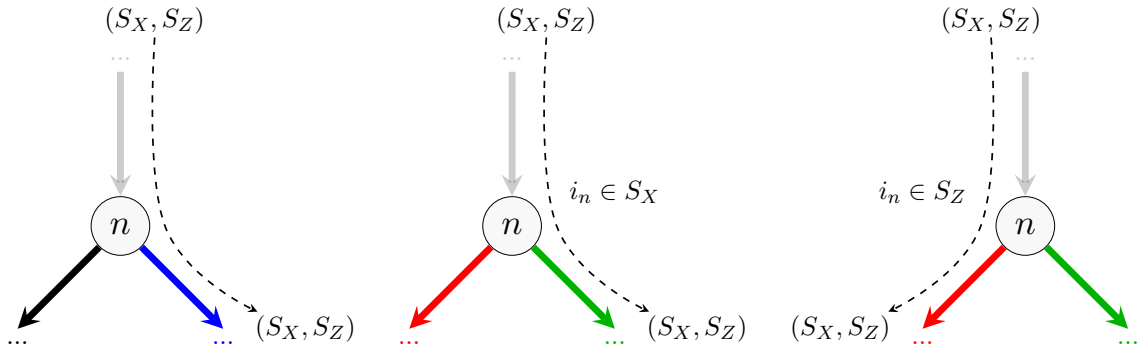
We therefore get a Shapley value $W(|S|, |S_{XZ}|)v_l = -W(|S_X|, |S_{XZ}|)v_l$.

■

The complexity reduction Theorem is the main reason Interventional TreeSHAP works so efficiently. Indeed, the exponential complexity of Shapley values w.r.t the number of input features used to severely limit their application to high-dimensional machine learning tasks. Interventional TreeSHAP gets rid of this exponential complexity by noticing that decision stumps h_P are `AND` functions. This drastically reduces the number of coalitions $S \subseteq S_{XZ} \setminus \{i\}$ to which adding player i changes the output of the model.

4.3 Efficient Tree Traversal

In this section, we leverage theoretical results from the previous section to improve the dynamic tree traversal in **Algorithm 1**. First, by virtue of **Theorem 13**, given the path P between the root and the current node in the traversal, we only need to keep track of the sets S_X and S_Z. Indeed once we reach a leaf (and P becomes a maximal path), we will only need these two sets of features to compute the Shapley values.



(a) **Scenario 1**: Ignore the type B edge and follow the type F edge without storing i_n.

(b) **Scenario 2**: Feature seen before in S_X so go down the direction of $\boldsymbol{x}$.

(c) **Scenario 2**: Feature seen before in S_Z so go down the direction of $\boldsymbol{z}$.

Figure 6

Now that we know the two data structures to track during the tree traversal, we are left with optimizing the tree traversal itself. **Corollary 8 & 10** give us sufficient conditions for when the contributions of line 3 of **Algorithm 1** are null. Hence, we must design the tree traversal in a way that avoids reaching maximal paths P whose contributions are guaranteed to be zero.

Firstly, from **Corollary 8**, we should avoid going down type B edges during the traversal. This is because any maximal path P that follows this edge will contain a type B edge and hence have null Shapley values. Therefore, if during the tree traversal we encounter a split where both $\boldsymbol{x}$ and $\boldsymbol{z}$ flow through the same edge, we only need to follow that edge of type F and the sets S_X and S_Z remain unchanged, see Figure 6(a). We call this the **Scenario 1**.

Secondly, **Corollary 10** informs us that, as we traverse the tree, we must only follow paths where the sets S_X and S_Z are disjoint. If we follow paths where the sets are not disjoint, then any maximal path we reach will have zero Shapley values resulting in wasted computations. Therefore, if a node n is encountered such that $\boldsymbol{x}$ and $\boldsymbol{z}$ go different ways, and the associated feature i_n is already in S_{XZ}, then

1. if $i_n \in S_X$, go down the same direction as $\boldsymbol{x}$
2. if $i_n \in S_Z$, go down the same direction as $\boldsymbol{z}$.

This is necessary to keep the sets S_X and S_Z disjoint. These two cases are illustrated in Figure 6(b) and (c) and are referred to as **Scenario 2**.

The final scenario that can occur when reaching a node is when $\boldsymbol{x}$ and $\boldsymbol{z}$ do not go the same way and the feature i_n is not already in S_{XZ}. In that case. we must visit both branches and update the sets S_X, S_Z accordingly, see Figure 7. We refer to this as the **Scenario 3**. We have now identified every scenario that can occur during the tree traversal, as well as the two data structures S_X, S_Z that we must store during the exploration. **Algorithm 2** presents the final pseudo-code for Interventional TreeSHAP based on **Theorem 13** and our insights on how to efficiently traverse the tree. The complexity of this algorithm depends

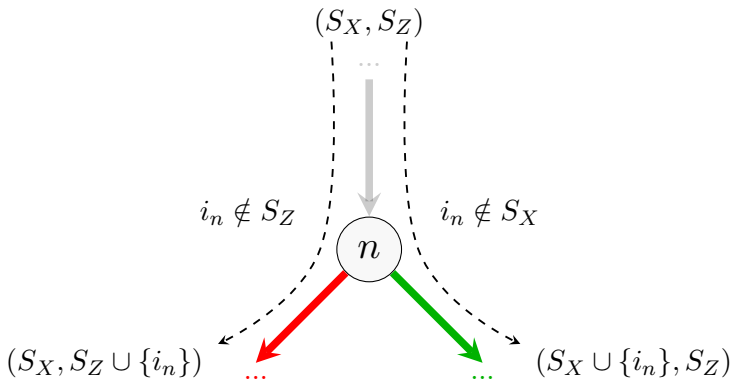


Figure 7: **Scenario 3**: Feature never seen before in S_X and S_Z so go down both directions and update S_X and S_Z accordingly.

on the data structure used to represent the sets S_X, S_Z. Interventional TreeSHAP must iterate through elements of S_X and S_Z in line 3. Moreover, on lines 4, 12, and 13, we check whether or not an index i_n belongs to one of these sets. If we encode the sets as binary vectors and keep track of their cardinality, the amount of computations per internal node is $\mathcal{O}(1)$ and is $\mathcal{O}(d)$ for terminal nodes (because of the for-loop at line 3). Hence, letting I be the set of internal nodes, we get a global complexity of $\mathcal{O}(|I| + |L|d)$. This is already a large improvement compared to the exponential complexity of classical Shapley values (assuming we have a tree whose number of nodes scales desirably w.r.t d).

Still, the current implementation of Interventional TreeSHAP leverages additional computation improvements to reduce the overall complexity to $\mathcal{O}(|I|) + \mathcal{O}(|L|) = \mathcal{O}(|N|)$. The intuition is that we can avoid the for-loop at line 3 by propagating the contributions at lines 5 and 7 up in the tree. However, we have found that this additional optimization does not generalize to other game theory indices, such as the Shapley-Taylor index presented in the coming section. Hence, we prefer to see **Algorithm 2** as the most general/versatile formulation of Interventional TreeSHAP.

Algorithm 2 Interventional Tree SHAP

procedure RECURSE(n, S_X, S_Z)
  if $n \in L$ **then**
    for $i \in S_X \cup S_Z$ **do** ▷ cf. Theorem 13
      if $i \in S_X$ **then**
        $\phi_i \mathrel{+}= W(|S_X| - 1, |S_{XZ}|)v_n$;
      else
        $\phi_i \mathrel{-}= W(|S_X|, |S_{XZ}|)v_n$;
      end if
    end for
  else if $\boldsymbol{x}_{\text{child}} = \boldsymbol{z}_{\text{child}}$ **then** ▷ Scenario 1
    return RECURSE($\boldsymbol{x}_{\text{child}}, S_X, S_Z$);
  else if $i_n \in S_X \cup S_Z$ **then** ▷ Scenario 2
    if $i_n \in S_X$ **then**
      return RECURSE($\boldsymbol{x}_{\text{child}}, S_X, S_Z$);
    else
      return RECURSE($\boldsymbol{z}_{\text{child}}, S_X, S_Z$);
    end if
  else ▷ Scenario 3
    RECURSE($\boldsymbol{x}_{\text{child}}, S_X \cup \{i_n\}, S_Z$);
    RECURSE($\boldsymbol{z}_{\text{child}}, S_X, S_Z \cup \{i_n\}$);
  end if
end procedure
$\boldsymbol{\phi} = \mathbf{0}$;
RECURSE(0, $S_X = \emptyset$, $S_Z = \emptyset$);
return $\boldsymbol{\phi}$;

5. Shapley-Taylor Indices

One of the purposes of deriving a proof for Interventional TreeSHAP is that it can be leveraged to discover efficient algorithms for computing other Game Theory indices. In this section, we employ previous results to prove how to compute Shapley-Taylor Indices efficiently.

5.1 Game Theory

Definition 14 (Shapley Taylor Indices (Sundararajan et al., 2020)) *Given a set of d players $[d] := \{1, 2, \dots, d\}$ and a cooperative game $\nu : 2^{[d]} \to \mathbb{R}$, the Shapley-Taylor Indices are defined as*

$$\Phi_{ij}(\nu) = \begin{cases} \nu(\{i\}) - \nu(\emptyset) & \text{if } i = j \quad (\textbf{\textit{Main Effect}}) \\ \sum\limits_{S \subseteq [d] \setminus \{i,j\}} W(|S|, d) \nabla_{ij}(S) & \text{otherwise} \quad (\textbf{\textit{Interactions}}). \end{cases} \tag{29}$$

with

$$\nabla_{ij}(S) = \nu(S \cup \{i, j\}) - \nu(S \cup \{j\}) - \big[\nu(S \cup \{i\}) - \nu(S)\big]. \tag{30}$$

The Shapley-Taylor Indices were invented as a way to provide interaction indices that respect the same additive property as the original Shapley values

$$\sum_{i=1}^{d} \sum_{j=1}^{d} \Phi_{ij}(\nu) = \nu([d]) - \nu(\emptyset). \tag{31}$$

As with classical Shapley values, the Shapley-Taylor indices assign no credit to dummy players. Letting D be the set of dummy players, we have

$$i \in D \;\text{ or }\; j \in D \Rightarrow \Phi_{ij}(\nu) = 0. \tag{32}$$

Like previously, the Shapley-Taylor indices of non-dummy players follow **Definition 14**, but where the set of all players $[d]$ is replaced by all non-dummy players D^C.

Lemma 15 (Dummy Reduction for Shapley-Taylor) *Let $D \subseteq [d]$ be the set of all dummy players of the game ν, and $D^C = [d] \setminus D$ be the set of non-dummy players, then*

$$\Phi_{ij}(\nu) = \begin{cases} 0 & \text{if } i \in D \text{ or } j \in D \\ \nu(\{i\}) - \nu(\emptyset) & \text{else if } i = j \\ \sum_{S \subseteq D^C \setminus \{i,j\}} W(|S|, |D^C|) \nabla_{ij}(S) & \text{otherwise.} \end{cases} \tag{33}$$

Proof The proof is mutatis mutandis like the proof of **Lemma 2**. ■

We consider in the sequel the Shapley-Taylor indices associated with the Baseline Interventional game $\nu_{h,\boldsymbol{x},\boldsymbol{z}}$ (cf. **Definition 3**) for a model h, an instance $\boldsymbol{x}$ and a baseline instance $\boldsymbol{z}$. We simply write $\boldsymbol{\Phi}(h, \boldsymbol{x}, \boldsymbol{z})$ in place of $\boldsymbol{\Phi}(\nu_{h,\boldsymbol{x},\boldsymbol{z}})$ for these indices.

5.2 Shapley-Taylor Indices Computation

Since the Shapley-Taylor Indices are also linear w.r.t coalitional games, computing these indices for a forest of decision trees reduces to the computation of these indices for a decision stump h_P associated with a maximal path P. Just like with Shapley values, we can use **Algorithm 1** as the skeleton of our algorithm. Fixing a decision tree h, instances $\boldsymbol{x}$ and $\boldsymbol{z}$, as well as a maximal path P in h, we focus on the computation of the Shapley-Taylor Indices $\boldsymbol{\Phi}(h_P, \boldsymbol{x}, \boldsymbol{z})$.

First we note that **Lemma 7 & 9** lead to similar corollaries about Shapley-Taylor Indices.

Corollary 16 *If P contains an edge of type B , then $\boldsymbol{\Phi}(h_P, \boldsymbol{x}, \boldsymbol{z}) = \mathbf{0}$.*

Proof Follows from **Lemma 7** and the linearity of Shapley-Taylor indices. ■

Corollary 17 *If $S_X \cap S_Z \neq \emptyset$, then $\boldsymbol{\Phi}(h_P, \boldsymbol{x}, \boldsymbol{z}) = \mathbf{0}$.*

Proof By **Lemma 9** and Linearity of the Shapley-Taylor values. ■

Once we reach a leaf we are meant to compute the Shapley-Taylor Indices of the decision stump h_P associated with the current maximal path P. We again use **Lemma 12** to reduce the complexity.

Theorem 18 (Complexity Reduction for Shapley-Taylor) *If P contains no type B edges and the sets S_X and S_Z are disjoint, then all features that are not in S_{XZ} are dummies and we get*

$$\Phi_{ii}(h_P, \boldsymbol{x}, \boldsymbol{z}) = h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\emptyset}(\boldsymbol{x})) \qquad i \in S_{XZ}, \tag{34}$$

$$\Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = \sum_{S \subseteq S_{XZ} \setminus \{i,j\}} W(|S|, |S_{XZ}|) \nabla_{ij}(S) \qquad i, j \in S_{XZ}, i \neq j \tag{35}$$

with

$$\nabla_{ij}(S) := h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S \cup \{i,j\}}(\boldsymbol{x})\,) - h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S \cup \{j\}}(\boldsymbol{x})\,) - \left[h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S \cup \{i\}}(\boldsymbol{x})\,) - h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x})\,)\right] \tag{36}$$

The exponential cost $\mathcal{O}(2^{|S_{XZ}|})$ of computing Φ_{ij} reduces to $\mathcal{O}(1)$ following

$$S_X = \{i\} \Rightarrow \Phi_{ii}(h_P, \boldsymbol{x}, \boldsymbol{z}) = v_l \tag{37}$$

$$i \in S_Z \text{ and } S_X = \emptyset \Rightarrow \Phi_{ii}(h_P, \boldsymbol{x}, \boldsymbol{z}) = -v_l \tag{38}$$

$$i \neq j \text{ and } i, j \in S_X \Rightarrow \Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = W(|S_X| - 2, |S_{XZ}|) v_l \tag{39}$$

$$i \neq j \text{ and } i, j \in S_Z \Rightarrow \Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = W(|S_X|, |S_{XZ}|) v_l \tag{40}$$

$$i \neq j \text{ and } i \in S_X, j \in S_Z \Rightarrow \Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = -W(|S_X| - 1, |S_{XZ}|) v_l \tag{41}$$

$$\text{else} \quad \Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = 0, \tag{42}$$

given that the coefficients W are computed and stored in advance.

Proof Equations (34) and (35) are a direct consequence of **Lemma 15**.

We will first tackle the coefficients $\Phi_{ii}(h_P, \boldsymbol{x}, \boldsymbol{z}) = h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\emptyset}(\boldsymbol{x}))$ with $i \in S_{XZ}$. There are only two possible cases where this difference is non-null. According to **Lemma 12**, when $S_X = \{i\}$, we have $h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\emptyset}(\boldsymbol{x})) = h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\emptyset}(\boldsymbol{x})) = v_l - 0 = v_l$. When $S_X = \emptyset$ and $i \in S_Z$, we have $h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{\emptyset}(\boldsymbol{x})) = h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X \cup \{i\}}(\boldsymbol{x})) - h_P(\boldsymbol{r}^{\boldsymbol{z}}_{S_X}(\boldsymbol{x})) = 0 - v_l = -v_l$. We have thus proven Equations (37) and (38).

Secondly, we tackle the non-diagonal elements $i \neq j$. To reduce the exponential complexity of computing these terms, we must identify the coalitions $S \subseteq S_{XZ} \setminus \{i, j\}$ for which

$$\nabla_{ij}(S) := h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i,j\}}(\boldsymbol{x})\,) - h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{j\}}(\boldsymbol{x})\,) - \big[h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S\cup\{i\}}(\boldsymbol{x})\,) - h_P(\, \boldsymbol{r}^{\boldsymbol{z}}_{S}(\boldsymbol{x})\,)\big]$$

is non-null. Recall that by **Lemma 12**, $h_P(\boldsymbol{r}^{\boldsymbol{z}}_S(\boldsymbol{x})) = 0$ for all $S \subseteq S_{XZ}$ except for $S = S_X$. So for the contribution of a coalition S to be non-zero, one of S, $S\cup\{i\}$, $S\cup\{j\}$ or $S\cup\{i,j\}$ must be equal to S_X. We distinguish four cases.

If $i, j \in S_X$, the only coalition $S \subseteq S_{XZ} \setminus \{i, j\}$ for which $\nabla_{ij}(S)$ is non zero is given by $S = S_X \setminus \{i, j\}$ and $\nabla_{ij}(S) = v_l - 0 - [0 - 0] = v_l$. Hence in this case, $\Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = W(|S|, |S_{XZ}|)v_l = W(|S_X| - 2, |S_{XZ}|)v_l$.

If $i, j \in S_Z$, and so $i, j \notin S_X$, the only coalition with a non-zero contribution is $S = S_X$ and $\nabla_{ij}(S) := 0 - 0 - \big[0 - v_l\big] = v_l$. It follows that $\Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = W(|S|, |S_{XZ}|)v_l = W(|S_X|, |S_{XZ}|)v_l$.

If $i \in S_X$ and $j \in S_Z$, the only coalition that contributes to the Shapley-Taylor indices is given by $S = S_X \setminus \{i\}$, which leads to $\nabla_{ij}(S) := 0 - 0 - \big[v_l - 0\big] = -v_l$. We have $\Phi_{ij}(h_P, \boldsymbol{x}, \boldsymbol{z}) = -W(|S|, |S_{XZ}|)v_l = -W(|S_X| - 1, |S_{XZ}|)v_l$.

Finally, the case where $j \in S_X$ and $i \in S_Z$ is symmetric and leads to the same formula as the previous case. This proves Equations (39), (40) and (41) and concludes the proof. ■

We present the procedure for efficient computations of Shapley-Taylor indices in **Algorithm 3**. Note that our results on Interventional TreeSHAP easily apply to these new indices, which we view as a testament to the great value of our mathematical proof for TreeSHAP.

Algorithm 3 Interventional Taylor-TreeSHAP

procedure RECURSE(n, S_X, S_Z)
  if $n \in L$ **then** ▷ cf. Theorem 18
    for $i \in S_X \cup S_Z$ **do**
      for $j \in S_X \cup S_Z$ **do**
        if $i = j$ **then**
          if $S_X = \{i\}$ **then**
            $\Phi_{ii} \mathrel{+}= v_n$;
          else if $S_X = \emptyset$ **then**
            $\Phi_{ii} \mathrel{-}= v_n$;
          end if
        else
          if $i, j \in S_X$ **then**
            $\Phi_{ij} \mathrel{+}= W(|S_X| - 2, |S_{XZ}|)v_n$;
          else if $i, j \in S_Z$ **then**
            $\Phi_{ij} \mathrel{+}= W(|S_X|, |S_{XZ}|)v_n$;
          else
            $\Phi_{ij} \mathrel{-}= W(|S_X| - 1, |S_{XZ}|)v_n$;
          end if
        end if
      end for
    end for
  else if $\boldsymbol{x}_{\text{child}} = \boldsymbol{z}_{\text{child}}$ **then** ▷ Scenario 1
    return RECURSE($\boldsymbol{x}_{\text{child}}, S_X, S_Z$);
  else if $i_n \in S_X \cup S_Z$ **then** ▷ Scenario 2
    if $i_n \in S_X$ **then**
      return RECURSE($\boldsymbol{x}_{\text{child}}, S_X, S_Z$);
    else
      return RECURSE($\boldsymbol{z}_{\text{child}}, S_X, S_Z$);
    end if
  else ▷ Scenario 3
    RECURSE($\boldsymbol{x}_{\text{child}}, S_X \cup \{i_n\}, S_Z$);
    RECURSE($\boldsymbol{z}_{\text{child}}, S_X, S_Z \cup \{i_n\}$);
  end if
end procedure
$\boldsymbol{\Phi} = \texttt{zeros}(d, d)$;
RECURSE(0, $S_X = \emptyset$, $S_Z = \emptyset$);
return $\boldsymbol{\Phi}$;

6. Extension to Partitions of Features

In certain settings, one may be interested in characterizing the effect of a set of features on the model's output, instead of considering their individual effects. This can be achieved by treating a set of features as a single super-feature. Assuming d' features are fed to the model, let us consider that we partition them in d groups: $[d'] = \bigcup\{P_i | i \in [d]\}$. This yields a new attribution problem where we wish to associate a real number to each group $i \in [d]$ that represents the contribution of the features in P_i, when considered as a group, toward the model output. An important application of group attributions is when one-hot encoded categorical features are fed to the Decision Trees. We will discuss this application later in the section.

A partition of features $[d']$ can be conveniently described with an indexing function $\mathcal{I} : [d'] \to [d]$ that associates each feature $j \in [d']$ to its group index $\mathcal{I}(j)$. Note that the pre-image map $\mathcal{I}^{-1} : 2^{[d]} \to 2^{[d']}$ allows to recover the groups of features using the partition notation: $\mathcal{I}^{-1}(\{i\}) = P_i$, $i \in [d]$. The partition of the d' features into d groups naturally induces a way to associate to any game ν on $[d']$ a game $\nu^{\mathcal{I}}$ on $[d]$ given by $\nu^{\mathcal{I}}(S) = \nu(\mathcal{I}^{-1}(S))$ for all $S \subseteq [d]$. In our specific case, given inputs $\boldsymbol{x}$ and $\boldsymbol{z}$ for a model h, we have

$$\nu^{\mathcal{I}}_{h,\boldsymbol{x},\boldsymbol{z}}(S) := \nu_{h,\boldsymbol{x},\boldsymbol{z}}(\mathcal{I}^{-1}(S)) = h(\, \boldsymbol{r}^{\boldsymbol{z}}_{\mathcal{I}^{-1}(S)}(\boldsymbol{x})\,). \tag{43}$$

See Figure 8 for an illustration of how the replace function $\boldsymbol{r}^{\boldsymbol{z}}_{\mathcal{I}^{-1}(S)}(\boldsymbol{x})$ is employed in $\nu^{\mathcal{I}}$. We note that features within the same group are always replaced simultaneously and so Shapley values for the game $\nu^{\mathcal{I}}_{h,\boldsymbol{x},\boldsymbol{z}}$ represent the joint contributions of each group towards the gap $h(\boldsymbol{x}) - h(\boldsymbol{z})$. Computing Shapley values for the game $\nu^{\mathcal{I}}_{h,\boldsymbol{x},\boldsymbol{z}}$ for a decision Tree h cannot be done using TreeSHAP, but we now show how to leverage our previous results to adapt TreeSHAP in this new setting.

Firstly, by linearity of the Shapley values we can again focus on a single maximal path P, and study the Shapley values of the game $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}$. Henceforth, we freely use the concepts and notations introduced in the previous sections for the computation of the Shapley values for the game $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}$ on $[d']$ and we show how to use them to analyze the game $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}$. For example, type X , type Z edges and the sets S_X, S_Z are all defined as in **Section 4** with respect to the maximal path h_P in a decision tree h and the inputs $\boldsymbol{x}$ and $\boldsymbol{z}$. We note that if P contains a type B edge, then **Lemma 7** ensures that the game $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}$ is null and this implies that the game $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}$ is null too. The resulting Shapley values $\boldsymbol{\phi}(\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}})$ are therefore constantly zero. Hence, we can again ignore maximal paths with type B edges. The next result is analogous to **Lemma 9** but for the game $\nu^{\mathcal{I}}$.

Lemma 19 *If $\mathcal{I}(S_X) \cap \mathcal{I}(S_Z) \neq \emptyset$, then*

$$\forall\, S \subseteq [d] \quad \nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S) = 0. \tag{44}$$

Proof Since $\mathcal{I}(S_X) \cap \mathcal{I}(S_Z) \neq \emptyset$, we can find an index $j \in [d]$ that belongs to both $\mathcal{I}(S_X)$ and $\mathcal{I}(S_Z)$. By definition of S_X and S_Z, this means that we can find in P an edge e of

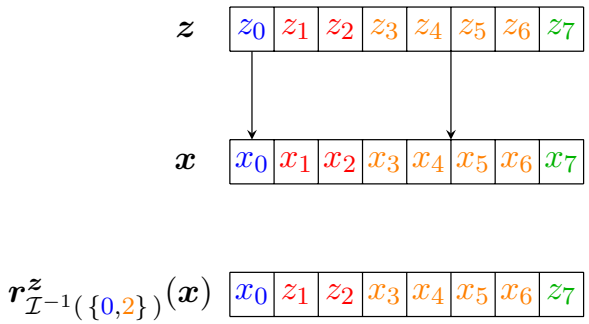


Figure 8: The replace function applied to partitions of $d' = 8$ features into $d = 4$ groups indicated by colors. Importantly, all features in the same group are replaced simultaneously.

X and an edge e' of type Z such that $\mathcal{I}(i_e) = j = \mathcal{I}(i_{e'})$. Now using Equation 20, for all $S \subseteq [d]$ we obtain:

$$\begin{aligned}\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S) = \nu_{h_P,\boldsymbol{x},\boldsymbol{z}}(\mathcal{I}^{-1}(S)) &\propto \mathbb{1}(i_e \in \mathcal{I}^{-1}(S))\mathbb{1}(i_{e'} \notin \mathcal{I}^{-1}(S)) \\ &= \mathbb{1}(\mathcal{I}(i_e) \in S)\mathbb{1}(\mathcal{I}(i_{e'}) \notin S) \\ &= \mathbb{1}(j \in S)\mathbb{1}(j \notin S) = 0.\end{aligned}$$

■

We have just identified two necessary conditions for the Shapley values of $\nu^{\mathcal{I}}$ to be null: if P contains a type B edge or if the sets $\mathcal{I}(S_X)$ and $\mathcal{I}(S_Z)$ are not disjoint. We will henceforth assume that none of these conditions apply to P so that the Shapley values are potentially non-null. We now wish to understand what players of $\nu^{\mathcal{I}}$ are dummies. The following Lemma establishes a relationship between dummies of ν and dummies of $\nu^{\mathcal{I}}$.

Lemma 20 *If $\mathcal{I}^{-1}(\{i\})$ is a set of dummies of ν, then i is a dummy player of $\nu^{\mathcal{I}}$. This is true for any games ν and $\nu^{\mathcal{I}}$ and more specifically Baseline Interventional Games $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}$.*

Proof First notice that if $D \subseteq [d']$ is a set of dummy players, then it easily follows by induction that for all $S' \subseteq [d'] \setminus D$ we have $\nu(S' \cup D) = \nu(S')$. Now let $S \subseteq [d] \setminus \{i\}$ and observe that when $\mathcal{I}^{-1}(\{i\})$ is a set of dummy players for ν we have:

$$\begin{aligned}\nu^{\mathcal{I}}(S \cup \{i\}) &= \nu\big(\mathcal{I}^{-1}(S \cup \{i\})\big) \\ &= \nu\big(\mathcal{I}^{-1}(S) \cup \mathcal{I}^{-1}(\{i\})\big) \\ &= \nu\big(\mathcal{I}^{-1}(S)\big) \\ &= \nu^{\mathcal{I}}(S)\end{aligned}$$

■

We now present an analogue to **Lemma 11**.

Lemma 21 *If i does not belong to $\mathcal{I}(S_{XZ})$, then it is a dummy player for the game $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}$.*

Proof If $i \notin \mathcal{I}(S_{XZ})$ then $\mathcal{I}^{-1}(\{i\}) \cap S_{XZ} = \emptyset$. So by **Lemma 11**, $\mathcal{I}^{-1}(\{i\})$ are dummies of $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}$ which by **Lemma 20** implies that i is a dummy feature of $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}$. ■

As only the features in S_{XZ} are not dummies of $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}$, it follows that

$$\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S) = \nu_{h_P,\boldsymbol{x},\boldsymbol{z}}(\mathcal{I}^{-1}(S) \cap S_{XZ}) \quad \text{for all } S \subseteq [d].$$

We will use this fact in the sequel. The following key Theorem shows how to efficiently compute the Shapley values of non-dummy players.

Theorem 22 (Complexity Reduction for Partitions) *If P contains no type B edges and the sets $\mathcal{I}(S_X)$ and $\mathcal{I}(S_Z)$ are disjoint, we get*

$$i \notin \mathcal{I}(S_{XZ}) \Rightarrow \phi^{\mathcal{I}}_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = 0 \tag{45}$$

$$i \in \mathcal{I}(S_X) \Rightarrow \phi^{\mathcal{I}}_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = W(|\mathcal{I}(S_X)| - 1, |\mathcal{I}(S_{XZ})|)v_l \tag{46}$$

$$i \in \mathcal{I}(S_Z) \Rightarrow \phi^{\mathcal{I}}_i(h_P, \boldsymbol{x}, \boldsymbol{z}) = -W(|\mathcal{I}(S_X)|, |\mathcal{I}(S_{XZ})|)v_l. \tag{47}$$

Proof The equation 45 is a direct consequence of **Lemma 21**. We now prove 46 and 47 separately.

Since all features in $[d] \setminus \mathcal{I}(S_{XZ})$ are dummies of $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}$, we can employ **Lemma 2** to rewrite the Shapley value definition

$$\phi_i(\nu^{\mathcal{I}}) = \sum_{S \subseteq \mathcal{I}(S_{XZ}) \setminus \{i\}} W(|S|, |\mathcal{I}(S_{XZ})|\,)\big(\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S \cup \{i\}) - \nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S)\big) \qquad i \in \mathcal{I}(S_{XZ}). \tag{48}$$

This summation contains an exponential number of terms, and like previous Theorems, only one of these terms will be non-zero. To identify which term is non-zero, we will again rely on **Lemma 12**. To do so, we note that since $\mathcal{I}(S_X)$ and $\mathcal{I}(S_Z)$ are disjoint, the sets S_X and S_Z must also be disjoint. Hence by **Lemma 12**, we have $\nu_{h_P,\boldsymbol{x},\boldsymbol{z}}(S') = 0$ for all $S' \subseteq S_{XZ}$ except for $S' = S_X$. Since as noted earlier $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S) = \nu_{h_P,\boldsymbol{x},\boldsymbol{z}}(\mathcal{I}^{-1}(S) \cap S_{XZ})$ for all $S \subseteq [d]$, **Lemma 12** ensures that we have the following: $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S) = 0$ for all $S \subseteq \mathcal{I}(S_{XZ})$ except when $\mathcal{I}^{-1}(S) \cap S_{XZ} = S_X$, or equivalently $S = \mathcal{I}(S_X)$. Therefore the only way a difference $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S \cup \{i\}) - \nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S)$ can be non-null is when either S or $S \cup \{i\}$ is equal to $\mathcal{I}(S_X)$.

Now, suppose $i \in \mathcal{I}(S_X)$. The only $S \subseteq \mathcal{I}(S_{XZ}) \setminus \{i\}$ for which the difference $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S \cup \{i\}) - \nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S)$ is not zero is when $S = \mathcal{I}(S_X) \setminus \{i\}$. For this set S, we get a contribution $W(|S|, |\mathcal{I}(S_{XZ})|)v_l = W(|\mathcal{I}(S_X) \setminus \{i\}|, |\mathcal{I}(S_{XZ})|)v_l = W(|\mathcal{I}(S_X)| - 1, |\mathcal{I}(S_{XZ})|)v_l$ thus proving Equation 46.

Finally, suppose $i \in \mathcal{I}(S_Z)$ and so $i \notin \mathcal{I}(S_X)$. Then the only way the difference $\nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S \cup \{i\}) - \nu^{\mathcal{I}}_{h_P,\boldsymbol{x},\boldsymbol{z}}(S)$ can be non-null is when $S = \mathcal{I}(S_X)$. For this set S, we get a contribution $-W(|S|, |\mathcal{I}(S_{XZ})|)v_l = -W(|\mathcal{I}(S_X)|, |\mathcal{I}(S_{XZ})|)v_l$ thus proving Equation 47. ■

We deduce from **Lemma 19** and **Theorem 22** that computing Shapley values for the game $\nu^{\mathcal{I}}$ can be achieved by substituting the sets $\mathcal{I}(S_X)$ and $\mathcal{I}(S_Z)$ to S_X and S_Z in the original TreeSHAP algorithm, as shown in Algorithm 4. The simplicity with which we were able to adapt TreeSHAP to partitions is a by-product of the theoretical results we obtained on TreeSHAP. We conclude by noting that if each feature is its own group (*i.e.* $d' = d$ and $\mathcal{I}(i) = i$), Algorithm 4 falls back to Algorithm 2. The coming subsection describes an application of Partition-TreeSHAP.

Algorithm 4 Interventional Partition-TreeSHAP

procedure RECURSE(n, $\mathcal{I}(S_X)$, $\mathcal{I}(S_Z)$)
  if $n \in L$ **then**
    for $i \in \mathcal{I}(S_X) \cup \mathcal{I}(S_Z)$ **do** ▷ cf. Theorem 13
      if $i \in \mathcal{I}(S_X)$ **then**
        $\phi_i \mathrel{+}= W(|\mathcal{I}(S_X)| - 1, |\mathcal{I}(S_{XZ})|)v_n$;
      else
        $\phi_i \mathrel{-}= W(|\mathcal{I}(S_X)|, |\mathcal{I}(S_{XZ})|)v_n$;
      end if
    end for
  else if $\boldsymbol{x}_{\text{child}} = \boldsymbol{z}_{\text{child}}$ **then** ▷ Scenario 1
    return RECURSE($\boldsymbol{x}_{\text{child}}, \mathcal{I}(S_X), \mathcal{I}(S_Z)$);
  else if $\mathcal{I}(i_n) \in \mathcal{I}(S_X) \cup \mathcal{I}(S_Z)$ **then** ▷ Scenario 2
    if $\mathcal{I}(i_n) \in \mathcal{I}(S_X)$ **then**
      return RECURSE($\boldsymbol{x}_{\text{child}}, \mathcal{I}(S_X), \mathcal{I}(S_Z)$);
    else
      return RECURSE($\boldsymbol{z}_{\text{child}}, \mathcal{I}(S_X), \mathcal{I}(S_Z)$);
    end if
  else ▷ Scenario 3
    RECURSE($\boldsymbol{x}_{\text{child}}$, $\mathcal{I}(S_X) \cup \mathcal{I}(\{i_n\})$, $\mathcal{I}(S_Z)$);
    RECURSE($\boldsymbol{z}_{\text{child}}$, $\mathcal{I}(S_X)$, $\mathcal{I}(S_Z) \cup \mathcal{I}(\{i_n\})$);
  end if
end procedure
$\boldsymbol{\phi} = \mathbf{0}$;
RECURSE(0, $\mathcal{I}(S_X) = \emptyset$, $\mathcal{I}(S_Z) = \emptyset$);
return $\boldsymbol{\phi}$;

6.1 Application to Feature Embedding

Classic training procedures for Decision Trees partition the input space at each internal node n via Boolean functions of the form $\mathbb{1}(x_{i_n} \leq \gamma_n)$. However, this model architecture assumes that feature values can be ordered, which always holds for numerical data, but not necessarily for categorical features *e.g.* $x_i \in [\text{Dog}, \text{Cat}, \text{Hamster}]$. In such scenarios, the model architecture itself must be adapted (Hastie et al., 2009, Section 9.2.4), or the categorical features must be embedded in a metric space during pre-processing (Guo and Berkhahn, 2016). An efficient approach for computing Interventional Shapley values in such contexts

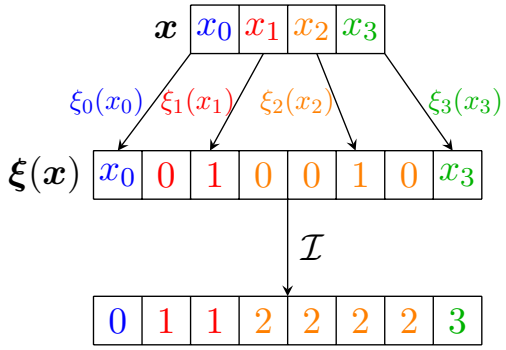


Figure 9: Example of a feature embedding $\boldsymbol{\xi}(\boldsymbol{x}) \in \mathbb{R}^8$. Here, x_0 and x_3 are kept intact while x_1 and x_2 are one-hot-encoded. The bottom of the figure presents the function $\mathcal{I}$ that maps the index of an embedded coordinate to the index of its associated $\boldsymbol{x}$ component.

depends on how categorical features are handled by the model. In this subsection, we show how Partition-TreeSHAP can compute Shapley values for categorical features when the latter approach is used. Namely, we consider here the case where the tree growth strategies are kept the same and categorical features are embedded before being fed to the model. This choice was made because `scikit-learn` Decision Trees do not yet natively support categorical features[3]. Concretely, assume that each feature $i \in [d]$ takes value in a set D_i which is assumed to be either $\mathbb{R}$ for numerical features or a finite set $[C]$ for a categorical feature taking C possible values. Before learning a tree, the data is preprocessed by choosing for every feature $i \in [d]$ an embedding function $\xi_i : D_i \to \mathbb{R}^{d_i}$. Several types of embedding are possible.

- The identity embedding $\xi_i(x_i) = x_i \in \mathbb{R}^1$ is used for numerical features where $D_i = \mathbb{R}$.
- The one-hot-encoding $\xi_i(x_i) = \boldsymbol{\delta}_{x_i} \in \mathbb{R}^C$ when x_i is categorical and takes C possible values, i.e. $D_i = [C]$. The components of the vector $\boldsymbol{\delta}_j$ are all 0 except for the jth component which is 1.
- The entity embedding $\xi_i(x_i) = \mathbf{E}[:, i] \in \mathbb{R}^{d_i}$ where the embedding matrix $\mathbf{E} \in \mathbb{R}^{d_i \times C}$ is learned *a priori* via a Neural Network (Guo and Berkhahn, 2016).

Letting $d' = \sum_{i=0}^{d} d_i$, we shall also define the global embedding function $\boldsymbol{\xi} : \mathbb{R}^d \to \mathbb{R}^{d'}$ that maps any given feature vector $\boldsymbol{x}$ to the concatenation $\boldsymbol{\xi}(\boldsymbol{x}) = [\xi_0(x_0), \ldots, \xi_d(x_d)]^T$. Figure 9 shows an example of embedding that relies on a mix of identity embeddings and one-hot encodings. For both training and prediction purposes, the vector $\boldsymbol{\xi}(\boldsymbol{x})$ stands as the input for the model in place of the feature vector $\boldsymbol{x}$. Therefore, applying TreeSHAP on the resulting Decision Trees will yield a separate value for each of the d' components in the embedded vector. However the d' features associated with the embedding lack interpretability and we argue it is more desirable to obtain a Shapley value for each d component of the original vector $\boldsymbol{x}$. Notice that our embedding $\boldsymbol{\xi}$ induces a surjective function $\mathcal{I} : [d'] \to [d]$

3. `https://github.com/scikit-learn/scikit-learn/pull/12866`

given by

$$\mathcal{I}(i) = \min \left\{ j \in [d] : i < \sum_{k=0}^{j} d_k \right\} \tag{49}$$

encoding a partition $\{P_i \mid i \in [d]\}$ of $[d']$ where each set P_i contains the indices of coordinates corresponding to feature i via the embedding $\boldsymbol{\xi}$. We therefore propose to use Partition-TreeSHAP with $\boldsymbol{\xi}(\boldsymbol{x}), \boldsymbol{\xi}(\boldsymbol{z})$ in place of $\boldsymbol{x}, \boldsymbol{z}$ and the mapping $\mathcal{I}$ of Equation 49. Doing so will provide a Shapley value for each of the d component of the original vector. Since feature embeddings are not supported in the current implementation of TreeSHAP from the SHAP library, we see Partition-TreeSHAP as a key contribution from a theoretical and practical perspective.

7. Conclusion

The Interventional TreeSHAP algorithm has previously revolutionized the computation of Shapley values for explaining decision tree ensembles. Indeed, before its invention, the exponential burden of Shapley values severely limited their application to Machine Learning problems, which typically involve large numbers of features. In this work, we take a step back and give a presentation of the mathematics behind the success of Interventional TreeSHAP. In doing so, our goal is both educational and research-oriented. On the one hand, the content of this paper can serve as a reference for a class on the mathematics behind interpretable Machine Learning. To this end, we also provide a simplified C++ implementation of Interventional TreeSHAP wrapped in Python as teaching material for the methods[4]. Our implementation is not meant to rival that of the SHAP library, which makes use of additional computational optimizations at the cost of clarity. On the other hand, we have shown that our theoretical results on Interventional TreeSHAP can be leveraged to effortlessly adapt the Shapley value computations to other game theory indices like Shapley-Taylor indices. Moreover, we have extended Interventional TreeSHAP to tasks where categorical features are embedded in a metric space before being fed to the model. We hope that our proof can stimulate future research in the application of game theory for post-hoc explanations of tree ensembles and similar models such as RuleFits.

Acknowledgements

The authors wish to thank the DEEL project CRDPJ 537462-18 funded by the National Science and Engineering Research Council of Canada (NSERC) and the Consortium for Research and Innovation in Aerospace in Québec (CRIAQ), together with its industrial partners Thales Canada inc, Bell Textron Canada Limited, CAE inc and Bombardier inc.

4. `https://github.com/gablabc/Understand_TreeSHAP`

References

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Hugh Chen, Scott M Lundberg, and Su-In Lee. Understanding shapley value explanation algorithms for trees. `https://hughchen.github.io/its_blog/index.html`, 2020. Accessed: 2022.

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

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.

Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable ai methods-a brief overview. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, pages 13–38. Springer, 2022.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1): 56–67, 2020.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, pages 307–317, 1953.

Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR, 2020.

Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The shapley taylor interaction index. In *International conference on machine learning*, pages 9259–9268. PMLR, 2020.