Understanding Machine learning models through Shapley values and Shapley residuals

Jason Ngo

December 17, 2020
Version: Fall Semester Final Draft
Advisor: Sorelle Friedler

Abstract

To explain black-box machine learning models, one popular method is to calculate the feature importance using Shapley value, a solution to distribute the payoff among the players in a coalition game. Though popular, Shapley-based feature importance methods suffer from interpretation issues when there are correlated features and out-of-distribution perturbed data points. Kumar et al. 2020b introduces *Shapley residuals* to quantify the information lost during the computation of Shapley values, e.g. feature interaction in the model. The exact computation of Shapley residuals is mathematically grounded but runs in exponential-time; the approximation computation is much faster but have not been verified heuristically. This thesis provides a paradigm that test the quality of the Shapley residual approximation algorithm through stress testing. By doing so, we hope to make clear, realistically, when or whether Shapley residuals can capture information that gets lost with Shapley-value based explanation methods.

Contents

1 Introduction

1.1 Why do we need interpretability?

Machine learning models have been widely adopted in the decision-making process in many critical domains; some uses include (i) determining whether a person receives loan from the bank, or (ii) modeling who is more likely to commit a crime (Goodman and Flaxman 2016). From these uses, it is clear that machine learning models have a large impact on people's lives; therefore, we must make sure these models are well understood and free of any biases or discrimination. This idea of understanding the machine learning models underlies our definition of machine learning model interpretability. Intuitively, the easier it is for someone to understand why certain decisions have been made, the more interpretable that machine learning model is.

1.2 What are the current approaches to interpretability?

Some *transparent* models like linear regression and decision tree are inherently easy to interpret because we can easily inspect the of each feature and know how each decision was made (cf. Figure 1.1). Other models like Neural Networks or Random Forest, however, are *opaque* and much harder to interpret (cf. Figure 1.2)[1]. This high complexity in a lot of machine learning models on one hand means they can be applied to many complicated tasks such as image classification, but on the other hand, it also means that there is a need for an interpretability method to understand these opaque models in a faithful and human-friendly manner.

To understand such opaque models, post-hoc interpretability methods, which aim to explain the model after the model is trained, are frequently used. Now, among the post-hoc interpretability methods, we are interested in the ones that are *model-agnostic*, which try to separate the explanations from the machine learning models. One advantage of model-agnostic interpretation methods over model-specific ones is

[1]For a more detailed description of these machine learning models, see Daumé III 2012.

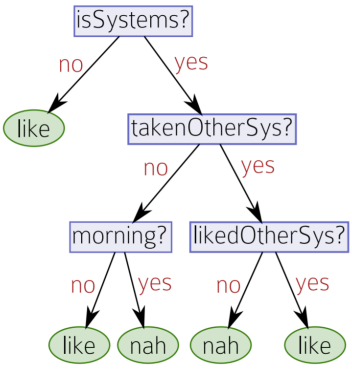

Fig. 1.1: A decision tree for a course recommender system (Daumé III 2012) that predicts whether a user will enjoy some course. In this figure, the questions are written in the internal tree nodes (rectangles) and the guesses are written in the leaves (ovals). We can work out the prediction by going through the branches of the tree and therefore can understand why the model makes such prediction.

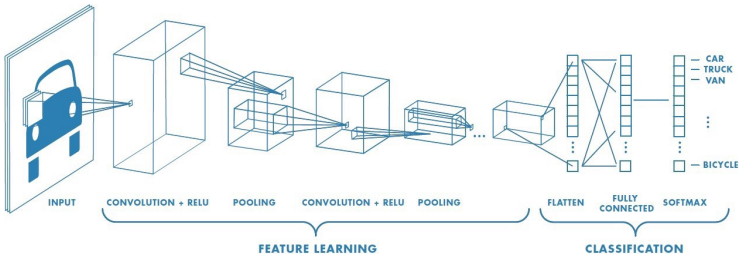

Fig. 1.2: A neural network architecture that predicts the label for an image (Patel and Pingel 2017). A neural network typically has many layers with a lot of parameters. It is not clear from this architecture how the model makes its prediction for an image of a car.

their flexibility: machine learning engineers are free to use any model for the task at hand (Molnar 2019). For example, to classify images, an intrinsically interpretable model like linear regression would not perform well, whereas a deep neural network would have much better performance; and the developers can use post-hoc model-agnostic method to study the behaviors of this deep neural network.

Model-agnostic techniques are designed to be generally applicable, depending only on the inputs and outputs of the model. The three most popular categories within post-hoc model-agnostic methods are: (a) model simplification, (b) feature importance, and (c) visualizations (Belle and Papantonis 2020). Model simplification methods aim to approximate the opaque model with a simpler model (say, decision tree, linear regression or "if-then" rules) that can explain the behavior of the opaque model locally around a data point (Ribeiro et al. 2016; Ribeiro et al. 2018). Feature importance methods aim to calculate the the contribution of each feature to the model output by permuting/transforming the feature set (see Lundberg and Lee 2017; Henelius et al. 2014). Finally, visualization methods aim to produce plots

like dependency plots or decision boundary plots that can explain how the opaque model makes decisions (see Goldstein et al. 2015).

1.3 Problem introduction

In the domain of chemical synthesis, a set of reactants is converted to one or more desired property descriptors through physical and chemical manipulations (for example, molecular weight is calculated from pH and polar surface area) (Raccuglia et al. 2016). Among these property descriptors, or *features* in a chemical synthesis machine learning model, we want to know which ones influence the model output the most.

In this thesis, I will focus on a feature importance method called SHAP, which uses game-theoretic Shapley value to calculate the average expected marginal contribution of each feature to the model's decision (Lundberg and Lee 2017). The SHAP method has many advantages: it has a solid theoretical foundation in game theory and it has a fast implementation for tree-based/neural network models (see Lundberg et al. 2018 for Tree SHAP and Lundberg and Lee 2017 for Deep SHAP). However, computations for a general machine model is slow (exponential time); and SHAP suffers from interpretation issues when there are correlated features and out-of-distribution perturbed data points (Kumar et al. 2020a; Slack et al. 2020a).

To address this problem, Kumar et al. 2020b introduces *Shapley residuals* to quantify the information about the machine learning model that get lost when the model is checked against SHAP. The exponential-runtime algorithm to calculate the Shapley residuals are given in Kumar et al. 2020b and have shown good preliminary results when applied to real-life datasets (cf. Section 2.3.2). However, there have not been any approximation algorithms or heuristics to test such algorithms.

This thesis, therefore, aims to provide such heuristics to test and verify Shapley residuals (more detail in Chapter 3). By doing so, we hope to make clear the information that gets lost with SHAP and how to apply both SHAP and Shapley residuals to understand the dataset of property descriptors.

2

Literature Review

This chapter introduces SHAP as a feature importance method based on the Shapley value from game theory. Then, the chapter presents some problems related to Shapley-value based and two current approaches, one of which is *Shapley residuals*, to quantify the uncertainty of Shapley-value based methods.

2.1 Shapley-value based methods

The Shapley-value based methods (Štrumbelj and Kononenko 2014; Lipovetsky and Conklin 2001; Lundberg and Lee 2017) fall under the category of feature importance interpretability methods, which aim to calculate the contribution of each feature to the model output. This idea of calculating model feature importance, conveniently, resembles the idea of Shapley value from game theory: given a machine learning model $f(x_1, x_2, \ldots, x_d)$, the features from 1 to d can be thought of as players in a coalition game.

2.1.1 Coalition game and Shapley value

Definition 2.1.1 (Coalition game, Kumar et al. 2020b)**.** A coalition game (or cooperative game) consists of d players and a value function $v : 2^{[d]} \to \mathbb{R}$ where $[d] = \{1, 2, \ldots, d\}$. The quantity $v(S)$ represents the value of the game for a coalition of players $S \subset 2^{[d]} := N$, with $v(\emptyset) = 0$.

The function v has the following meaning: if S is a coalition of players, then $v(S)$ is the worth of that coalition, describing the total expected sum of payoffs the members of S can obtain by cooperation. In the interpretability context, for a coalition of features S, $v(S)$ is the model output given only the features in S.

Definition 2.1.2 (Shapley value, Shapley 1965)**.** The Shapley value is one solution to distribute the coalition worth among players. The amount of payoff[1] that player i gets given a coalition game $(v, [d])$ is:

$$\phi_i(v) = \sum_{S \subseteq [d]} \frac{|S|!(d - |S| - 1)}{d!} \left(v(S \cup \{i\}) - v(S)\right). \tag{2.1}$$

The formula can be interpreted as follows: imagine the coalition being formed one at a time, with each player i demanding $v(S \cup \{i\}) - v(S)$ as a fair compensation. Then, the total payoff will be the average of the compensations taken over all possible permutations in which the coalition can be formed. In the interpretability context, the Shapley value $\phi_i(v)$ for a machine learning model v can then be viewed as the influence of feature i on the outcome.

Properties 2.1.3 (Shapley 1965). The Shapley value satisfies these properties:

- Efficiency: the sum of the Shapley values for all players in coalition equals the worth of that coalition: $\sum_{i=1}^{d} \phi_i(v) = v(N)$.
- Dummy: the Shapley value for a null player (someone who does not contribute) is 0: if $v(S \cup \{i\}) = v(S)$ for all $S \subset N \setminus \{i\}$, then $\phi_i(v) = 0$.
- Symmetry: the Shapley values for two equivalent players are the same: if $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subset N \setminus \{i, j\}$, then $\phi_i(v) = \phi_j(v)$.
- Linearity: the worth of a combined game should equal the distributed worth of the component games: if v, v' are two games on d players, then $\phi_i(\alpha v + \alpha' v') = \alpha\phi_i(v) + \alpha'\phi_i(v')$.

Example 2.1.4 ((Kumar et al. 2020b)). Suppose we have the following two coalition games (or machine learning models) with three players (or features), where the players are independent and identically distributed on $[0, 1]$. The value functions for the games are:

$$f_1(x_1, x_2, x_3) = x_1 + x_2 + x_3,$$
$$f_2(x_1, x_2, x_3) = x_1 + 2x_2x_3.$$

Note that $f_1(1, 1, 1) = f_2(1, 1, 1) = 3$. Let us calculate the Shapley values for the three players. With three players in the game, we have 8 possible coalitions: order them $S_{0,0,0}, S_{0,0,1}, \ldots, S_{1,1,1}$, with the indices representing whether the player x_i is present in the coalition or not.

[1] In this thesis, we only focus on the payoff for individual players. In practice, we can also calculate the payoff for any subset of players $S \subseteq [d]$.

To calculate the Shapley value for player x_1 in the game f_1, we will calculate the sum of marginal contribution of adding x_1 to the four coalitions $\{S_{0,0,0}, S_{0,1,0}, S_{0,0,1}, S_{0,1,1}\}$ that do not yet contain x_1

$$\begin{aligned}\phi_1(f_1) &= \frac{0!(d-1)!}{d!}(v(S_{1,0,0}) - v(S_{0,0,0})) + \frac{1!(d-2)!}{d!}(v(S_{1,1,0}) - v(S_{0,1,0})) \\ &\quad + \frac{1!(d-2)!}{d!}(v(S_{1,0,1}) - v(S_{0,0,1})) + \frac{2!(d-3)!}{d!}(v(S_{1,1,1}) - v(S_{0,1,1})) \\ &= \frac{1}{3}(1) + \frac{1}{6}(1) + \frac{1}{6}(1) + \frac{1}{3}(1) \\ &= 1.\end{aligned}$$

Now, by similar calculations, we can see that the Shapley values for three players in both games are all 1, meaning they each get the same payoff from the coalition worth. In other words, these three players all contribute an equal amount to both games.

In the next section, we will explore a method that uses Shapley value to explain a machine learning model.

2.1.2 SHapley Additive exPlanations (SHAP)

SHAP is a Shapley-value based method proposed by Lundberg and Lee 2017 and built on the works of Štrumbelj and Kononenko 2014; Lipovetsky and Conklin 2001 to explain individual predictions of a machine learning model. Given a machine learning model $f(x_1, x_2, \ldots, x_d)$, the features from 1 to d can be thought of as players in a coalition game. The Shapley value $\phi_i(f)$ can then be viewed as the influence of feature i on the outcome.

Additive feature attribution

Given f as the original prediction model and x as the instance to be explained. Local explanation models like SHAP use a coalition vector[2] x' that map to the original inputs through a mapping function $x = h_x(x')$, and then uses an explanation model g that tries to ensure $g(z') \approx f(h_x(x'))$ whenever $z' \approx x'$ (Note that x' may contain less information than x and that h_x is specific to the current input x) (Ribeiro et al. 2016). In the SHAP method, the Shapley value explanation model g is represented as an additive feature attribution method:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{2.2}$$

[2]In the original thesis Lundberg and Lee 2017, "simplified input" is used instead of "coalition vector". This thesis uses the latter term to be consistent with our game-theoretic treatment of Shapley value.

where $z' \in \{0,1\}^M$ is the coalition vector that can be mapped back to the original input x through a mapping function h_x, M is the maximum coalition size, and $\phi_j \in \mathbb{R}$ is the Shapley value for a feature j. In the coalition vector, a value of 1 or 0 means that the corresponding feature is "present" or "absent" respectively. To approximate the output $f(x)$ of the original model, we set the coalition vector to be a vector of 1's (meaning all features are present), and ϕ_0 to be the model output with the coalition vector set to 0. In this case, $z' = x'$ and $g(z') = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i$.

Properties 2.1.5 (Lundberg et al. 2018). SHAP describes the following three desirable properties for local explanation methods and proves that the additive feature attribution method in Equation (2.2) is the only method that satisfies all three properties:

1. Local accuracy: when approximating the original model f for a specific input x, local accuracy requires the explanation model g to match the output of f for the coalition vector x':

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

   Here, if we set the coalition vector x' to be all 1's, then this equation is the same as the Shapley efficiency property in Section 2.1.3.

2. Missingness: the features missing in the coalition vector must have no impact:

$$x'_i = 0 \implies \phi_i = 0$$

3. Consistency: if a model changes so that the marginal contribution of a feature value increases or stays the same regardless of other features, then the Shapley value for that feature should also increase or stay the same: Let $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ denote setting $z'_i = 0$ in the coalition vector. For any two models f and f', if

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i)$$

   for all inputs $z' \in \{0,1\}^M$, then $\phi_i(f', x) \geq \phi_i(f, x)$.

KernelSHAP

Now that we have seen the additive (or linear) explanation model g in Equation (2.2) and its properties, this section introduces KernelSHAP (Lundberg and Lee 2017), an algorithm that tries to find the explanation model g.

Let f be the model we are trying to explain, g be a linear explanation model, $\pi_{x'}(z')$ be the weighting kernel that measures the similarity between two coalition

vectors, and $\Omega(g)$ be the regularization term that measures complexity of g. The local interprability method aims to find the linear model g that best approximates f around the given instance x.

Definition 2.1.6 (KernelSHAP, Lundberg and Lee 2017)**.** KernelSHAP aims find a linear model of Shapley values:

$$\mathcal{E} = \arg \min_{\text{linear model } g} L(f, g, \pi_{x'}) + \Omega(g),$$

where

$$\begin{aligned} \Omega(g) &= 0, \\ \pi_{x'}(z') &= \frac{M-1}{\binom{M}{|z'|}|z'|(M-|z'|)}, \\ L(f, g, \pi_{x'}) &= \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_{x'}(z'), \end{aligned}$$

with $|z'|$ being defined as the number of non-zero elements in the coalition vector $z' \in Z$ (the set of all coalition vectors). The coefficients of this linear model are the Shapley values (same as Equation (2.2)).

Remark 2.1.7. Consider the weighting kernel $\pi_{x'}(z')$. This function reaches its maximum when $|z'| \in \{1, M-1\}$, i.e. the weight for the coalition that consists of a single feature or all but one feature is the largest. In these cases, we can learn about the features' isolated main effect on the prediction. If a coalition consists of half the features, we cannot learn about an individual feature's contribution, as there are many possible coalitions with half the features; in this case, the weight for this coalition would be the smallest (Molnar 2019).

Remark 2.1.8. Let S be the set of features corresponding to the non-zero indices in the coalition vector z'. Some alternate notations for the expected model output are:

$$f(h_x(z')) = f_x(z') = v_{f,x}(S). \tag{2.3}$$

The first and second notations were introduced in Lundberg and Lee 2017, while the third notation was used in Kumar et al. 2020a to stay consistent with the game-theoretic treatment.

2.2 Problems with SHAP and other Shapley-value based methods

While SHAP has a solid theoretical foundation in game theory and a fast model-specific implementation, the computations for the model-agnostic framework is slow. Specifically, for a machine learning model with d features, there are 2^d coalitions to calculate the payout for (hence exponential time). In addition to this inefficient runtime, SHAP suffers from interpretation issues when there is feature dependence, in which case KernelSHAP might put too much weight on those features (Kumar et al. 2020a); or when there is out-of-distribution perturbed data points (Slack et al. 2020a).

2.2.1 Issues with correlated features

For a model with an arbitrarily large feature set, we cannot use KernelSHAP to calculate the payout for all the 2^d coalitions; rather, we must only consider the coalitions that are meaningful to understanding the model. The problem then becomes which features should be chosen in the cooperative game, and which features are redundant, for the choice of *correlated* features could greatly influence the Shapely values (as can be seen in the following example).

Example 2.2.1 (Shapley value for redundant feature). Consider the following function $f(x_A, x_B, x_C)$ with two identical features B and C, given as a table below. Since these two features are identical, we know that $v_{f,x}(\{B\}) = v_{f,x}(\{C\}) = v_{f,x}(\{B, C\})$ and $v_{f,x}(\{A, B\}) = v_{f,x}(\{A, C\}) = v_{f,x}(\{A, B, C\})$. Let $f'(x_A, x_B) = f(x_A, x_B, x_C)$. For any data instance, since $x_B = x_C$, we have $f(\mathbf{x}) = f'(\mathbf{x})$. Now, let's calculate the Shapley values for $\phi_A(f)$, $\phi_B(f)$, $\phi_A(f')$, and $\phi_B(f')$.

A	B	C	f
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	3

Tab. 2.1: Function with redundant feature C, whose values exactly match those of feature B. The model output f is given for each combination of (A, B) values.

10

Using the same calculation process as outlined in Equation (2.1) and Example 2.1.4, we have:

$$\phi_A(f) = \frac{5}{3} = 1.67, \qquad \phi_B(f) = \frac{2}{3} = 0.67,$$
$$\phi_A(f') = 1.5, \qquad \phi_B(f') = 1.5.$$

Here, we can see that if we include feature C, the Shapley value for A is larger for B (meaning feature A is more important than B); whereas if we do not include feature C, the Shapley values for A and B are the same (they are equally important).

This example highlights an interpretation issue with the SHAP approach: if the Shapley value for a feature is relatively low, it might mean that the feature is unimportant, but it could also mean that a strong univariate effect is being diluted among correlated features.

2.2.2 Issues with conditional expectation

To estimate the model output with only features in $S \subset [d]$ are known, one can fix the values of the features in S, and integrate over the product of the marginal distribution (denoted as $\mathcal{D}$) of the features *not* in S:

$$v_{f,x}(S) = E_{\mathcal{D}}[f(x_S, \mathbf{X}_{\bar{S}})]. \tag{2.4}$$

Unfortunately, this distribution $\mathcal{D}$ can be very different from the distribution of the original data; and since a machine learning model can perform differently for in-distribution and out-of-distribution data, SHAP's approach of permuting the set of features can be *gamed* (Slack et al. 2020a).

Suppose an adversary would want to deploy a biased classifier f into the real world. This biased classifier will be tested against SHAP to detect whether f is biased. Let $\mathcal{X}_{dist}$ be the distribution of real world data, and $\mathcal{X}$ be the set of data points that the adversary has access to. By exploiting the fact that most perturbed data points from SHAP are out of distribution, the adversary could bypass the test by designing an *adversarial classifier* e that exhibits biased behavior on instances sampled from $\mathcal{X}_{dist}$, and remain unbiased on instances that do not come from $\mathcal{X}_{dist}$.

Let ψ be an unbiased classifier, the adversarial classifier e will take the form:

$$e(x) = \begin{cases} f(x), & \text{if } x \in \mathcal{X}_{dist} \\ \psi(x), & \text{otherwise.} \end{cases}$$

In order to build such adversarial classifier, an adversary need to be able to decide whether a data point is in or out of distribution $\mathcal{X}_{dist}$, which can easily be done by

11

training an off-the-shelf model on $\mathcal{X}$ and $\mathcal{X}_p := \{\text{all perturbations of instances in } \mathcal{X}\}$, which are assigned labels "in-distribution" and "out-of-distribution" respectively.

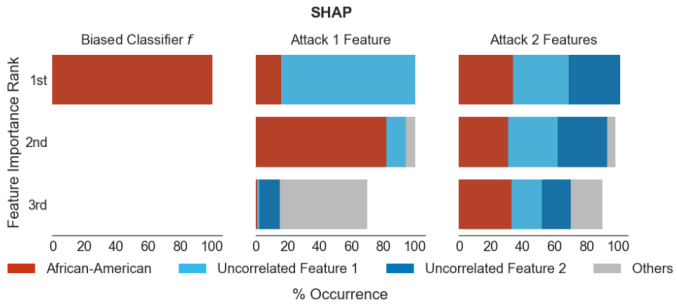

Fig. 2.1: Adversarial attack on SHAP: Experimental result on the COMPAS dataset (Slack et al. 2020a). In the first plot, we can see that the classifier f is clearly biased as it based 100% of its predictions on the African-American feature. However, when tested with SHAP (2nd and 3rd plots), African-American feature were shown to be much less important in the prediction process, making the biased classifier f appear innocent.

2.3 Current approaches to quantify uncertainty in SHAP explanations

Given the SHAP method and its problems presented in Sections 2.1-2.2, we want to tackle these problems by quantifying the uncertainty in SHAP explanations which can come from (1) representing machine learning model as a cooperative game, (2) fitting a linear explanation model locally around a data point, and/or (3) approximating Shapley values. In this section, we focus on the uncertainty associated with local explanation model and with Shapley value through the two approaches: Bayesian Local Explanation and Shapley residuals.

2.3.1 Bayesian Local Explanation

Recognizing that the SHAP explanation method is not stable (nearly identical input can yield vastly different output), Slack et al. 2020b proposes Bayesian Local Framework (and BayesianSHAP) to estimate the point-wise estimates of feature importance as credible intervals (see Figure 2.2).

Bayesian Local Framework uses a generative process and inference procedure to capture two sources of explanation uncertainty: uncertainty associated with the feature importance score ϕ, and the uncertainty over the error term $\epsilon \sim \mathcal{N}\left(0, \frac{\sigma^2}{\pi_x(z)}\right)$

12

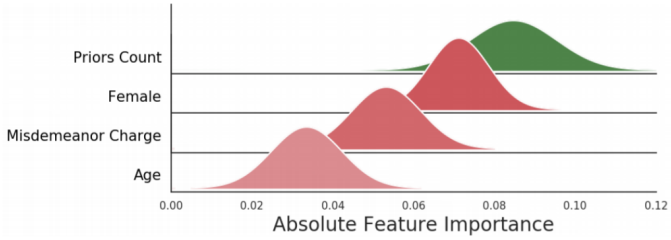

Fig. 2.2: Bayesian Local Framework applied to the COMPAS dataset. Green indicates features that positively contribute to the prediction, while red indicates otherwise. For this particular prediction, `Priors Count` and `Female` seem to be the most important features, with uncertainty about their relative importance depicted in the graph. (Slack et al. 2020b)

that arises due to the mismatch between the local explanation model g and the local decision surface of the black box model f.

To compute the uncertainty for ϕ, repeated sampling from ϕ's posterior distribution is used to get 95% confidence intervals.

To compute the uncertainty over the error term ϵ, we first calculate ϵ's marginal posterior distribution and evaluate the probability density function of this posterior at 0, i.e. $P(\epsilon = 0)$ to estimate how well g matches f. Note that these computations are possible because the posterior distribution on ϕ and σ^2 follow normal and inverted-chi-square distributions, respectively. Details of the derivations for these distributions are omitted due to complexity but they can be found in (Slack et al. 2020b).

2.3.2 Shapley residuals

Besides the above Bayesian local explanation framework, Shapley residuals by Kumar et al. 2020b is another approach that tries to quantify the limit of SHAP. In this thesis, Shapley values are interpreted as the orthogonal projection from 2^d-dimensional space to d-dimensional space; and Shapley residuals captures information lost in this projection process. For example, experimental result on the NHANES dataset with Shapley residuals shows that Shapley residuals can tell practitioners about feature interaction that is not shown when the model is analyzed with only SHAP (cf. Figure 2.6).

To understand Shapley residuals, we will (1) represent cooperative game on a hypercube, (2) introduce the definition of inessential games, and (3) introduce Shapley residuals that captures how much a cooperative game is far from being inessential.

Hypercube representation

Definition 2.3.1 (Hypercube representation, Stern and Tettenhorst 2019)**.** Given the set of players $[d] = N$, define the oriented graph $G = (V, E)$ by

$$V = 2^N, \qquad E = \{(S, S \cup \{i\}) \in V \times V : \ S \subset N \setminus \{i\}, i \in N\}.$$

This is precisely the d-dimensional hypercube graph, where each vertex corresponds to a coalition $S \subset N$, and where each edge corresponds to the addition of a single player $i \notin S$ to S, oriented in the direction of the inclusion $S \hookrightarrow S \cup \{i\}$.

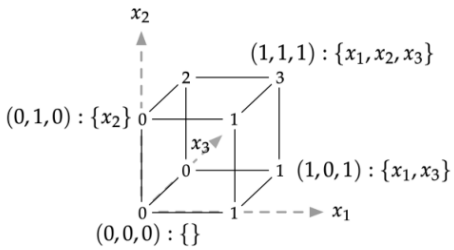

Fig. 2.3: Graphical representation of $f_2(x_1, x_2, x_3) = x_1 + 2x_2x_3$ from Example 2.1.4 (Kumar et al. 2020b)

Let $\mathbb{R}^V$ be the real-valued functions from V, and $\mathbb{R}^E$ be the real-valued functions from E.[3] The differential operator $d : \ \mathbb{R}^V \to \mathbb{R}^E$ is then defined as:

$$dv(S, S \cup \{i\}) = v(S \cup \{i\}) - v(S).$$

Intuitively, d takes in two adjacent vertices and assigns the contribution of the added feature to the edge between those two vertices (see Figure 2.4).

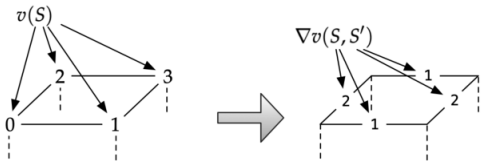

Fig. 2.4: Graphical representation of d operator on f_2 from (2.1.4) (Kumar et al. 2020b)

Finally, we will define a partial gradient $d_i : \ \mathbb{R}^V \to \mathbb{R}^E$:

$$d_i v(S, S \cup \{i\}) = \begin{cases} v(S \cup \{j\}) - v(S), & i = j \\ 0, & \text{otherwise.} \end{cases}$$

[3] One can show that both $(\mathbb{R}^V, \|\cdot\|_2)$ and $(\mathbb{R}^E, \|\cdot\|_2)$ are normed vector spaces.

d_i only evaluates a gradient for edges corresponding to the insertion of feature i.

Inessential game decomposition

Definition 2.3.2 (Inessential game, Stern and Tettenhorst 2019)**.** The game v is inessential if $v(S) = \sum_{i \in S} v(\{i\})$ for all $S \subset N$. That is, each coalition obtains the same value working together as its individual members would obtain working separately.

Example 2.3.3. Going back to our two cooperative games introduced in Example 2.1.4:

$$f_1(x_1, x_2, x_3) = x_1 + x_2 + x_3,$$
$$f_2(x_1, x_2, x_3) = x_1 + 2x_2x_3,$$

we can see that f_1 is inessential because the value for any coalition S is the sum of the contribution of its members, whereas f_2 is not inessential because

$$f_2(0, 1, 1) \neq f_2(0, 1, 0) + f_2(0, 0, 1).$$

From this example, we can see that inessentiality captures information about the model that Shapley value calculations does not.

Proposition 2.3.4 (Stern and Tettenhorst 2019)**.** *The game v is inessential if and only if for each $i \in [d]$, there exists $v_i \in \mathbb{R}^V$ such that $d_i v = dv_i$. In other words, a game v being inessential is equivalent to the partial gradients of v being the full gradient for some other game.*

A decomposition of an arbitrary game v into games that are "close to being inessential" follows. Since v is not guaranteed to be inessential, there might not exist v_i such that $dv_i = d_i v$; however, we can find the closest such v_i by solving the least squares problem:

$$v_i = \arg\min_{x \in \mathbb{R}^V} \|d_i v - dx\|$$

Stern and Tettenhorst 2019 shows that the Shapley value for player ith is the same as the worth of $v_i(N)$, where N is the set of all players (see Figure 2.5).

Shapley residuals

From the decomposition above, Kumar et al. 2020b proposes *Shapley residuals* to quantify the degree of deviation from inessentiality.

Definition 2.3.5 (Shapley residuals, Kumar et al. 2020b)**.** We call $r_i = d_i v - dv_i$ the Shapley residual of player i. Analogously, $r_S = \sum_{i \in S} r_i$ is the Shapley residual of S.

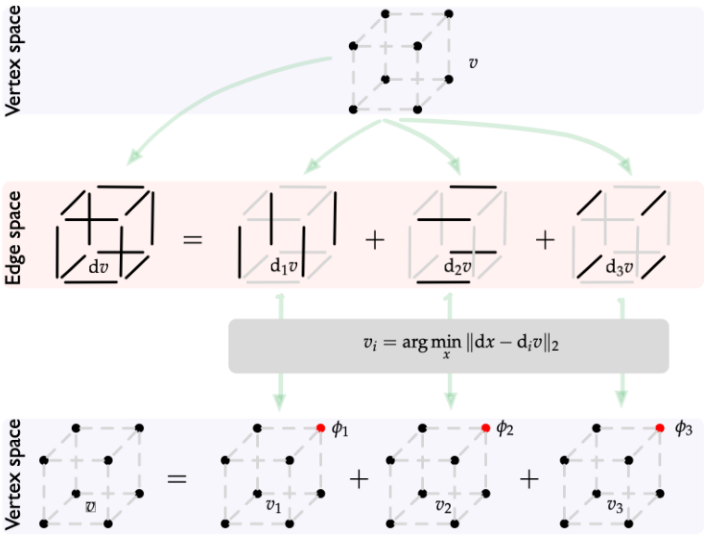

Fig. 2.5: Decomposition of a game proposed by Stern and Tettenhorst 2019; Figure by Kumar et al. 2020b. Given a game v, calculate the partial gradients $d_i v$ for the edge space. For each partial gradient $d_i v$, we will find another game v_i that minimizes the least square problem. The worth for the grand coalition of v_i (marked in red) is the same as the Shapley value $\phi_i(v)$.

Example 2.3.6. For $f_1(x_1, x_2, x_3) = x_1 + x_2 + x_3$, the Shapley residuals are $\|r_1\| = \|r_2\| = \|r_3\| = 0$ because this game is inessential. However, for $f_2(x_1, x_2, x_3) = x_1 + 2x_2x_3$, using the game decomposition above we obtain:

$$\|r_1\|^2 = 0,\ \|r_2\|^2 = \|r_3\|^2 = 2.$$

From this example, we can see that Shapley residuals show the existence of interacting terms between x_2 and x_3 that SHAP alone fails to show.

As interaction between variables increases, the residual value $\sum_i \|r_i\|^2$ increase and the SHAP values deviate further and further from the coefficients in the actual model (see Figure 2.6 for an example of this concept in practice).

(a) KernelSHAP values

(b) KernelSHAP residuals

Fig. 2.6: Shapley values and residuals on an XGBoost mortality model for age and sex (Kumar et al. 2020b). We notice that the importance of `sex` feature is stable across age ranges: males have a higher risk of mortality. However, the residuals show that middle-aged men may have other interacting factors that contribute to their mortality predictions. Shapley residuals, therefore, can provide practitioners with information previously lost by the SHAP method.

3

Future Work Proposal: Testing the quality of Shapley residuals

In the chemical synthesis context, since the set of property descriptors is highly correlated, in order to understand feature importance, we need *both* SHAP and Shapley residuals. The exact computation for Shapley residuals, however, is exponential, making it not viable for the chemical dataset with a lot of features. An approximation algorithm for Shapley residuals exists and is introduced below, but we need to test the quality of such algorithm via stress testing.

3.1 Shapley residuals Approximation Algorithm

An $(1+\epsilon)$-approximation algorithm for Shapley residuals was therefore developed by my research group. To estimate a residual $\|r_i\|^2$, the algorithm first samples at random an entry of r_i and then returns an estimate of that entry with provable error bounds. By doing so, the algorithm provides an estimate for $\|r_i\|^2$ (see Algorithm 1).

Algorithm 1: Estimating $\|r_i\|^2$.

$\hat{r} = 0$

for τ *iterations* **do**

  Pick random edge $e \in E$

  $\hat{r} = \hat{r} + \texttt{Estimate}^2(e, v)$

end

return $2^d \hat{r}/\tau$

Recall that $r_i = d_i v - d v_i$. To estimate an entry of r_i, the algorithm estimates v_i with the use of sub-Gaussian and sub-exponential random variables. Now, to estimate $\|r_i\|^2$ as a whole, it gives bounds for the total magnitude $\|v_i\|^2$ by looking at the spectral bounds of the solution operator to our least square problem; this method works because any inessential games are eigenvectors corresponding to the smallest non-zero eigenvalues of this solution operator. Given the bounds of $\|v_i\|$ and en estimate for entries in r_i, the algorithm then returns an estimate of $\|r_i\|^2$.

18

3.2 Proposed stress testing paradigm

Although this approximation has been proven mathematically to be within $(1+\epsilon)$-bound of Shapley residuals, there is currently no testing paradigm to test the quality of this algorithm. For example, if we scale up the number of variables, we want to know whether the approximation of Shapley residual is still informative of the machine learning model. This thesis, therefore, wishes to provide with such paradigm through stress testing to answer two questions: (1) does the Shapley residual approximation match the feature interaction in the dataset/model?; and (2) when does Shapley residual provide insight as opposed to interpretation noise?

To do so, I will come up with (1) synthetic datasets of various sizes (in terms of number of instances and features), feature interaction level, and type (regression versus classification); as well as (2) machine learning models of various types and interaction terms between the parameters.

The approximations of Shapley residuals are then computed in a *controlled* manner (all those properties of the model/dataset can be tuned with a hyperparameter), checked against the nature of our synthetic data for quality assessment. For instance, for a given machine learning model, if we increase the feature interaction in the dataset, the Shapley residuals should also increase.

Through stress testing, we hope to test the quality of the Shapley residual approximation algorithm and make clear when or whether Shapley residuals can capture information that gets lost with SHAP through the approximation algorithm.

4

Conclusion

In Chapter 2 Literature Review, we have seen how Shapley value, a solution to distribute the payoff among the players in a coalition game, can be used to explain a machine learning model. In this context, the features are "players" and their importance is the respective "payoff". Based on Shapley value, SHAP by Lundberg and Lee 2017 assigns a feature importance value ϕ_i to each feature x_i for a particular prediction by utilizing (1) an additive feature importance method, and (2) theoretical results verifying the desired properties of this method.

Although it has a solid theoretical foundation, SHAP (and other Shapley-based value methods) suffer from exponential runtime and interpretation issues when there are correlated features and out-of-distribution perturbed data points (Kumar et al. 2020a; Slack et al. 2020a). To address this problem, Slack et al. 2020b uses a generative model to measure the uncertainty of each feature importance value and uncertainty of the explanation model's quality. Taking on a similar approach, using the hypercube representation of coalition game, Kumar et al. 2020b introduces *Shapley residuals* as a measure of feature interaction to quantify the information lost during the computation of Shapley values. The interpretation of Shapley residuals is built on solid mathematical foundation and has shown positive preliminary results (cf. Figure 2.6) as a way to quantify the uncertainty of SHAP.

Computation for Shapley residuals, however, is exponential. While an approximation algorithm exists, we need to apply stress testing in a controlled manner to test the quality of that algorithm. Practitioners of both SHAP and Shapley residuals should not adopt an unskeptical use of these methods until the algorithms for both are fully verified.

20

References

Belle, Vaishak and Ioannis Papantonis (2020). "Principles and Practice of Explainable Machine Learning". In: *arXiv preprint arXiv:2009.11698* (cit. on p. 3).

Daumé III, Hal (2012). "A course in machine learning". In: *Publisher, ciml. info* 5, p. 69 (cit. on pp. 2, 3).

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin (2015). "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation". In: *Journal of Computational and Graphical Statistics* 24.1, pp. 44–65 (cit. on p. 4).

Goodman, Bryce and Seth Flaxman (2016). "EU regulations on algorithmic decision-making and a "right to explanation"". In: *ICML workshop on human interpretability in machine learning (WHI 2016), New York, NY. http://arxiv. org/abs/1606.08813 v1* (cit. on p. 2).

Henelius, Andreas, Kai Puolamäki, Henrik Boström, Lars Asker, and Panagiotis Papapetrou (2014). "A peek into the black box: exploring classifiers by randomization". In: *Data mining and knowledge discovery* 28.5-6, pp. 1503–1529 (cit. on p. 3).

Kumar, I Elizabeth, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler (2020a). "Problems with Shapley-value-based explanations as feature importance measures". In: *Proceedings of the International Conference on Machine Learning 1 pre-proceedings (ICML 2020)* (cit. on pp. 4, 9, 10, 20).

Kumar, IE, C Scheidegger, S Venkatasubramanian, and S Friedler (2020b). "Shapley Residuals: Quantifying the limits of the Shapley value for explanations." In: *ICML Workshop on Workshop on Human Interpretability in Machine Learning (WHI)* (cit. on pp. ii, 4–6, 13–17, 20).

Lipovetsky, Stan and Michael Conklin (2001). "Analysis of regression in game theory approach". In: *Applied Stochastic Models in Business and Industry* 17.4, pp. 319–330 (cit. on pp. 5, 7).

Lundberg, Scott M, Gabriel G Erion, and Su-In Lee (2018). "Consistent individualized feature attribution for tree ensembles". In: *arXiv preprint arXiv:1802.03888* (cit. on pp. 4, 8).

Lundberg, Scott M and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 4765–4774. URL: `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf` (visited on 02/10/2020) (cit. on pp. 3–5, 7–9, 20).

Molnar, Christoph (2019). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. `https://christophm.github.io/interpretable-ml-book/` (cit. on pp. 3, 9).

Patel, Shyamal and Johanna Pingel (2017). *Introduction to Deep Learning: What Are Convolutional Neural Networks?* URL: `https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html` (cit. on p. 3).

Raccuglia, Paul, Katherine C Elbert, Philip DF Adler, Casey Falk, Malia B Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A Friedler, Joshua Schrier, and Alexander J Norquist (2016). “Machine-learning-assisted materials discovery using failed experiments”. In: *Nature* 533.7601, pp. 73–76 (cit. on p. 4).

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). “" Why should I trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144 (cit. on pp. 3, 7).

– (2018). “Anchors: High-Precision Model-Agnostic Explanations.” In: *AAAI*. Vol. 18, pp. 1527–1535 (cit. on p. 3).

Shapley, Lloyd (1965). *Notes on N-Person Games VII. Cores of Convex Games*. Tech. rep. RAND CORP SANTA MONICA CALIF (cit. on p. 6).

Slack, Dylan, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju (2020a). “Fooling lime and shap: Adversarial attacks on post hoc explanation methods”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186 (cit. on pp. 4, 10–12, 20).

Slack, Dylan, Sophie Hilgard, Sameer Singh, and Himabindu Lakkaraju (2020b). “How Much Should I Trust You? Modeling Uncertainty of Black Box Explanations”. In: *arXiv preprint arXiv:2008.05030* (cit. on pp. 12, 13, 20).

Stern, Ari and Alexander Tettenhorst (2019). “Hodge decomposition and the Shapley value of a cooperative game”. In: *Games and Economic Behavior* 113, pp. 186–198 (cit. on pp. 14–16).

Štrumbelj, Erik and Igor Kononenko (2014). “Explaining prediction models and individual predictions with feature contributions”. In: *Knowledge and information systems* 41.3, pp. 647–665 (cit. on pp. 5, 7).