Quantifying uncertainties in feature importance measures for machine learning models

---

Ruiming Li

November 20, 2020
Version: Full Draft
Advisor: Sorelle Friedler

Abstract

In this thesis, I study an approach inspired by game theory to explain how features in the input data affect the output of machine learning models without access to the models' internal working. This approach assigns a number called Shapley value to each feature to indicate its contribution to the model output, but has a few limitations and uncertainties. I investigate three sources of uncertainty of Shapley value and respective methods to quantify the uncertainty, using Shapley Residuals to capture missing information in the game theory representation, Mean-Standard-Error to quantify the sampling error in Shapley value estimation, and Bayesian SHAP to calculate the statistical variations in SHAP explanation model. I aim to evaluate their combined effect on the trust-worthiness of Shapley explanations for real-life models. My goal is to make machine learning models more interpretable to humans so we can gain meaningful knowledge from them.

Acknowledgements

I would like to thank professor Sorelle Friedler for all of her guidance as my advisor throughout the semester. I am really grateful for professor Sara Mathieson's detailed advice and Jason Ngo's collaboration on the topic. Last but not the least, this thesis would not be possible without all the support from my friends and family. Thank you all for making it possible.

Contents

1

Introduction

1.1 Motivation

Machine learning models are trained to perform accurately on a task by learning from existing data. As more sophisticated models such as neural networks are used to achieve higher accuracy, they also become increasingly difficult for humans to interpret and understand their decision mechanism. We call these complex models whose internal workings are obscure to human understanding *black box*.

The lack of interpretability for black box models can lead to practical and ethical concerns (Molnar 2019). Passengers want to understand how self-driving cars process vision information and make decisions before they can trust the cars. Banks may use a machine learning model to grant loans for millions of people, but if the model learns biases from existing data and grants loans based on the applicant's race, we will perpetuate racial biases if we do not understand and improve the model. In scientific discoveries, machine learning models are applied to make predictions on the outcome of chemical experiments and expedite material discoveries (Raccuglia et al. 2016). Despite the models' high accuracy, chemists still need to be able to interpret the models to gain more knowledge about the mechanism behind chemical reactions.

As a result, we want to explain black box models in a human interpretable manner, so humans can understand what the models are doing with input data, and what are driving the decisions for the output.

1.2 Previous Work

1.2.1 Machine Learning

Machine learning is a set of methods that computers use to make and improve predictions or behaviors based on data (Molnar 2019). It has three major components: a representation of data we want to learn from, a loss function to quantify the difference between machine prediction and actual data, and an optimization method to minimize the loss function.

Supervised learning is a type of machine learning where the data comprises of features and labels. A feature is an individual measurable property or description of a data instance, such as the size of a house. A label is a property we want to make predictions on, such as the price of a house. A supervised machine learning model is essentially a function that maps a set of features to labels. The goal of supervised learning is to find such function given a set of known features and labels, so the difference between machine output and actual label (measured by loss function) is minimized in a process called training. After training, we can then apply the model to make predictions for new, unlabelled data in a processed called testing.

One supervised machine learning model is decision tree, where every node in the tree is a feature and every branch is a feature value, and each leaf is a predicted label. For example, we can predict the number of rented bikes on a certain day with a decision tree, using the number of days since 2011 to take into account the general trend and the temperature of the specific day:

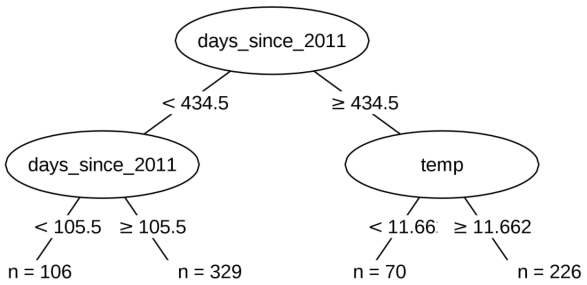

Fig. 1.1: Decision tree model where each circle is a feature, each line is feature value, and each leaf is the predicted label, number of rented bikes. (Molnar 2019)

1.2.2 Interpretability

To understand machine learning models, one approach is to use simple, inherently interpretable machine learning models directly. In the field of supervised learning, decision trees and k-nearest neighbors are themselves human interpretable. We can understand a decision tree by traversing all nodes in the tree and looking at the decision variable and value at every node (Fig. 1.1). However, these models are limited in their use cases since they may not be accurate and appropriate for all tasks.

A more general approach to explanation should be model-agnostic. With a model-agnostic interpretation, we can apply it to any type of machine learning model without access to its internal working. One of the approaches is to explain by simplification. We can use simple models such as decision trees to approximate the decisions made by complex black box models and interpret the simple approximation model instead. The greatest limitation of this approach is that the simple model may not be flexible enough to accurately model the complex model (Belle and Papantonis 2020).

Instead of constructing an approximation model to the entire black box, we can instead focus on explaining how a decision is made for a specific data point. This is a local explanation approach since we are not explaining the global black-box model, but individual data instances. A proposed method called LIME (Local Interpretable Model-agnostic Explanations) works by training the best local surrogate model around a data instance that approximates the performance of the black box model around the same local region (Ribeiro et al. 2016). It draws data instances around a desired explanation region and construct a linear model to approximate the behaviors of the black box (Fig. 1.2).

A recent approach called SHAP (SHapley Additive exPlanations) is inspired by game theory and similarly explains how a decision is made for a specific data instance (Lundberg and Lee 2017). SHAP is a feature relevant explanation approach, which quantifies each feature's effect on the output prediction (Fig. 1.2). It draws inspiration from game theory by considering each feature as a player and the prediction task as a game. Each feature's individual contributions to the total game outcome shows the feature's importance in the black box model when predicting on a data instance. In game theory, such contribution is quantified by the Shapley value.

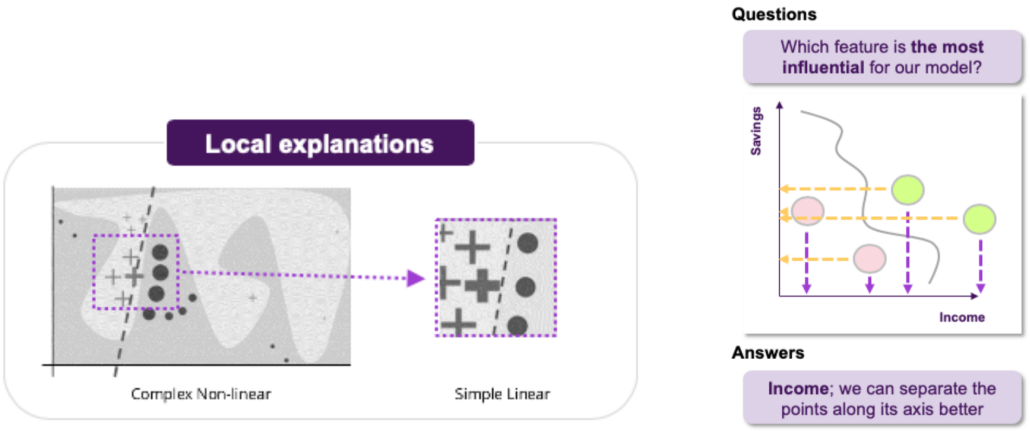

Fig. 1.2: Left: local explanation methods (LIME) Right: feature importance methods (SHAP) (Belle and Papantonis 2020)

1.3 Problem Introduction

This thesis will focus on exploring the benefits and drawbacks of using Shapley value as a feature importance measure for interpretability, and finding ways to better apply Shapley value in black box explanation.

Proponents of Shapley value argue that it is the best feature importance measure since Shapley value is based on the solid mathematical foundation of game theory, which distributes the difference between an instance's prediction and the average prediction fairly among all feature values, while other methods such as LIME does not guarantee that.

However, many also point out the limitations of the Shapley value. It is computationally expensive (taking exponential time) to calculate the Shapley values exactly. Existing approximation methods to Shapley value suffer from mathematical inconsistencies (I Elizabeth Kumar et al. 2020), adversarial vulnerabilities (Slack, Hilgard, Jia, et al. 2020), and human-centric explanation issues (I Elizabeth Kumar et al. 2020).

To better understand and apply Shapley value, I. Kumar et al. 2020 proposed a new idea of Shapley residuals, which quantifies the limits of the Shapley value for explanations. Shapley residual is a measure of the information loss when we map a high dimensional game to Shapley value, a scalar value for each feature. On

the other hand, Slack, Hilgard, Singh, et al. 2020 and Merrick and Taly 2020 use statistical methods to quantify the uncertainty as a confidence interval when we draw samples to estimate an explanation model. We aim to study three uncertainties measurement methods and better understand the question of how much we can trust Shapley value explanations as a feature importance measure.

2

Literature Review

2.1 Shapley Value

In game theory, Shapley value is a numerical measure of players' contribution in a cooperative game. A game is defined by a set of players $S \subseteq \{1, 2, \cdots, N\}$ and a value function $v : 2^{[N]} \to \mathbb{R}$ that maps a subset of the players to real numbers. The value function represents the value the set of players generate when they cooperate in the game (Molnar 2019).

The value of the grand coalition $v(\{1, 2, \cdots, N)\}$ is the value achieved in a game if all players cooperate. It represents the collective payoff from the game in total. Shapley value aims to quantify the individual payoff for each player i in the game: how much does i contribute to the coalition?

The marginal contribution $\Delta_v(i, S)$ of player i with respect to a coalition S is the additional value, measured by value function v, that is generated with player i's cooperation in the game:

$$\Delta_v(i, S) = v(S \cup i) - v(S)$$

The Shapley value for player i in the entire game is the weighted average of its marginal contribution with respect to all possible subset of players. Hence, the Shapley value ϕ_v of player i is given by:

$$\phi_v(i) = \sum_{S \subseteq \{1, \cdots, N\} \setminus \{i\}} \frac{|S|!(N - |S| - 1)!}{N!} \Delta_v(i, S)$$

7

We can understand the weight for each subset by looking at the possible ordering for N players. In total, there are $N!$ random orderings for all players. Out of all orderings, we are interested in those associated with the value function $\Delta_v(i, S)$, the marginal contribution of player i with respect to the subset S. This means we should first have all players in S included in the game and immediately followed by player i, a situation where i is added to subset S. So far, there are $|S|!$ ways to order the players in S. After players in S and player i, there are still $N - |S| - 1$ players left, so we can arrange them in $(N - |S| - 1)!$ ways, thus giving rise to the term $\frac{|S|!(N-|S|-1)!}{N!}$ for every $\Delta_v(i, S)$.

2.2 Shapley Value as Feature Importance Measure

To apply Shapley value for feature importance measures, we can consider the features of the input data as players in a cooperative game. The value of the grand coalition when all the features work together is the model prediction at a data instance. We can split up the total prediction value to marginal contribution of individual features, thus quantifying each feature's importance.

For example, consider a linear model with two independent features $f(x_1, x_2) = x_1 + 2x_2$. We seek to explain the outcome $f(1, 1) = 3$. The marginal contribution of x_1, x_2 with respect to possible subsets are:

$$\Delta_v(x_1, \{\}) = v(\{\} \cup x_1) - v(\{\}) = f(x_1 = 1, x_2 = 0) - f(x_1 = 0, x_2 = 0) = 1$$

$$\Delta_v(x_1, \{x_2\}) = v(\{x_2\} \cup x_1) - v(\{x_2\}) = f(x_1 = 1, x_2 = 1) - f(x_1 = 0, x_2 = 1) = 1$$

$$\Delta_v(x_2, \{\}) = v(\{\} \cup x_2) - v(\{\}) = f(x_1 = 0, x_2 = 1) - f(x_1 = 0, x_2 = 0) = 2$$

$$\Delta_v(x_2, \{x_1\}) = v(\{x_1\} \cup x_2) - v(\{x_1\}) = f(x_1 = 1, x_2 = 1) - f(x_1 = 1, x_2 = 0) = 2$$

Hence, using the formula, we can calculate the Shapley value for x_1 and x_2:

$$\phi_v(x_2) = \frac{(0!(2 - 0 - 1)!)}{2!} \cdot 2 + \frac{(0!(2 - 1 - 1)!)}{2!} \cdot 2 = 2$$

$$\phi_v(x_1) = \frac{(0!(2 - 0 - 1)!)}{2!} \cdot 1 + \frac{(0!(2 - 1 - 1)!)}{2!} \cdot 1 = 1$$

In the above example, we made the assumption that the expected value $E(x_1) = E(x_2) = 0$ so the absence of a feature x_i from the set S is indicated by setting $x_i = 0$, and the value function $v(S)$ with respect to S is equal to $f(S, x_i = 0 \; \forall x_i \notin S)$. This

is because we cannot evaluate a model without some features, such as inputting the empty subset $\{\}$ to f to evaluate the respective value function. Instead, we replace the value for features not in the set by the expected value of these features to get the expected output of f without these features. In practice, there are two ways to calculate the value function with a subset of features in the model: the conditional and interventional approaches. They both aim to find the expected value of f over the features not in set S.

Let X be a multivariate random variable $\{X_1, X_2, \cdots, X_N\}$, and X_S be the set of random variables in the subset of features S: $X_S = \{X_i : i \in S\}$. Let x be a set of actual values for all input features $\{x_1, \cdots, x_N\}$, and x_s be a set of values of the features in S: $x_s = \{x_i : i \in S\}$. Given a model f and all input feature values x, the conditional value function when only the features in S are known is the expected model output given all values of the features in S (I Elizabeth Kumar et al. 2020):

$$v_{f,x}(S) = E[f(X)|X_s = x_s]$$

$$= \int \cdots \int P(X_{\bar{S}}|X_s = x_s)P(X_s = x_s)f(X_s = x_s, X_{\bar{S}})\, dX_1 \cdots dX_j,\ X_j \in \bar{S}$$

This means that we are simply setting all values of $X_s = x_s$ and integrate over the remaining conditional distribution for features not in S ($\bar{S}$).

In the interventional approach, instead of conditioning on the set of feature values first, we simply create a joint distribution between feature in the set X_S and all features not in S and calculate the expected model output $f(X_S = x_S, X_{\bar{S}})$. When we set the known features values $X_S = x_S$, we can find the joint distribution by multiplying all marginal distributions of features in $\bar{S}$ (I Elizabeth Kumar et al. 2020):

$$v_{f,x}(S) = E[f(X_S = x_S, X_{\bar{S}})]$$

$$= \int \cdots \int P(X_{\bar{S}})P(X_s = x_s)f(X_s = x_s, X_{\bar{S}})\, dX_1 \cdots dX_j,\ X_j \in \bar{S}$$

2.3 Shapley Value Estimation

Calculating Shapley value exactly is usually impractical, due to the exponential number of coalitions. One approximation technique for Shapley value is through Monte-Carlo sampling. Suppose we want to explain the model output $f(\mathbf{x})$ where $\mathbf{x}$

is an input data instance, a vector of size N. Our goal is to approximate the Shapley value for each feature x_i with some collection of data instances $\mathbf{X}$.

For example, we can approximate Shapley values $\phi(x_1), \phi(x_2)$ to explain the model output $f([1\ 1]) = 3$. While we calculated the Shapley values exactly in section 2.2 given the model $f(x_1, x_2) = x_1 + 2x_2$ and assuming $E(x_1) = E(x_2) = 0$, we do not have access to the equation of f anymore and do not have knowledge of the expected feature value. Instead, we are given available input data instances $\mathbf{X}$, such as $\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \end{pmatrix}$. Note that we have no knowledge of but can still call the model f on any given input data $\mathbf{x}$.

The Shapley value $\phi(x_j)$ for a feature j is the weighted sum of its marginal contribution with respect to all possible subsets of features. To approximate the marginal contribution of feature j, we draw a random instance $\mathbf{z}$ from $\mathbf{X}$ and separately generate a random order of the features. We order $\mathbf{x}$ and $\mathbf{z}$ by the random ordering: $\mathbf{x}_o = (x_{(1)}, \ldots, x_{(j)}, \ldots, x_{(p)})$ and $\mathbf{z}_o = (z_{(1)}, \ldots, z_{(j)}, \ldots, z_{(p)})$. We then create two new instances by combining values from the instance of interest x and the sample z: one new instance is considered as with feature j: $\mathbf{x}_{+j} = (x_{(1)}, \ldots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$ and the other is considered as without j: $\mathbf{x}_{-j} = (x_{(1)}, \ldots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$. We then compute the difference of the model output as the marginal contribution $\Delta_v(x_j) = f(\mathbf{x}_{+j}) - f(\mathbf{x}_{-j})$. Consider drawing M such instances and calculating the marginal contribution $\Delta_v^m(x_j)$ for each time. Averaging over all M times gives an approximation of the Shapley value: $\phi(x_j) = \frac{1}{M} \sum_{m=1}^{M} \Delta_v^m(x_j)$. The averaging process implicitly weighs samples by the probability distribution (Molnar 2019).

The following algorithm describes the above process:

Algorithm 1 Approximating Shapley estimation for single feature value

Require: Number of iterations M, instance of interest $\mathbf{x}$, feature index j, data matrix X, and machine learning model f

procedure APPROXIMATESHAPLEYVALUE
  for $i = 1$ to M **do**
    Draw $\mathbf{z}$ from X
    Choose a random permutation o of the feature values
    $\mathbf{x}_o = (x_{(1)}, \ldots, x_{(j)}, \ldots, x_{(p)})$
    $\mathbf{z}_o = (z_{(1)}, \ldots, z_{(j)}, \ldots, z_{(p)})$
    $\mathbf{x}_{+j} = (x_{(1)}, \ldots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
    $\mathbf{x}_{-j} = (x_{(1)}, \ldots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
    $\Delta_v(x_j) = f(\mathbf{x}_{+j}) - f(\mathbf{x}_{-j})$
  end for
  return $\phi(x_j) = \frac{1}{M} \sum_{m=1}^{M} \Delta_v^m(x_j)$
end procedure

2.4 SHapley Additive exPlanations (SHAP)

Shapley values are numerical measurements of feature importance. To explain machine learning model around a data instance, we can construct a simplified, linear explanation model around the instance of interest using Shapley values. SHAP is one such model that applies Shapley value explanation to construct an additive feature attribution model (Lundberg and Lee 2017).

Our goal is to explain a machine learning model f with a linear explanation model g. We can map an individual input instance $\mathbf{x}$ for f to a coalition vector $\mathbf{z}' \in \{0,1\}^N$, such that a value of 1 or 0 in $\mathbf{z}'_i$ means that the corresponding feature i is present or absent respectively. A SHAP explanation model g is a linear combination of features' Shapley values:

$$g(z') = \phi_0 + \sum_{i=1}^{N} \phi_i z'_i$$

where $\phi_i \in \mathbb{R}$ is the Shapley value for feature i. To approximate the model output at $f(\mathbf{x})$, we can set $z' = \mathbf{x}'$ to be a vector of all 1's meaning all features are present, and set $\phi_0 = g(\mathbf{0}) = E(f)$ meaning the expected model output for the entire data distribution. This gives us the model $f(\mathbf{x}) = g(\mathbf{x}') = \phi_0 + \sum_{i=1}^{N} \phi_i$, where the complex model output is explained by a linear model that sums the Shapley value

11

for each feature and the expected model output for the entire distribution (Lundberg and Lee 2017).

2.5 Problems with Shapley value

2.5.1 Mathematical Issues

Conditional Value Function

The conditional approach to estimating model output (section 2.2) for a subset of features suffers from one major mathematical issue: it attributes influence to features with no interventional effect (I Elizabeth Kumar et al. 2020). Interventional effect measures how much the model output changes by intervening on the values of a feature alone. For a feature $i \in \{1, 2, \cdots, N\}$, if $f(\mathbf{x}) = f(\mathbf{x}')$ whenever $x_j = x'_j$ for all $j \neq i$, meaning if we can get the same model output f with other features except i, then feature i does not affect the function output at all and has no interventional effect.

This can happen when feature i is highly correlated with other features, so the value of i can be predicted with some other features and the model may make i's interventional effect negligible. However, in the conditional value function, since we conditioned on a set of known features before taking the expected value of the model output, a non-interventional feature i can affect the expected value if i is correlated to some features not in S.

The question then arises: should we consider features with no interventional effect as input to the function? If two features are highly correlated, should we consider them one of the same feature or two separate features? We can illustrate this effect with a dataset with redundant features (I Elizabeth Kumar et al. 2020). Suppose a dataset has three features A, B, C, where $P(X_B = X_C) = 1$, meaning all values of B and C are the same for each data instance. Consider a model $f(X_A, X_B, X_C)$. The value functions for the game with three features X_A, X_B, X_C are:

$$\phi_v(A) = \frac{1}{3}\Delta_v(A, \{\}) + \frac{1}{6}\Delta_v(A, \{B\}) + \frac{1}{6}\Delta_v(A, \{C\}) + \frac{1}{3}\Delta_v(A, \{B, C\})$$

$$\phi_v(B) = \frac{1}{3}\Delta_v(B, \{\}) + \frac{1}{6}\Delta_v(B, \{A\}) + \frac{1}{6}\Delta_v(B, \{C\}) + \frac{1}{3}\Delta_v(B, \{A, C\})$$

Since $P(X_B = X_C) = 1$, by our conditional definition of value function, we know 1) the value functions for A with respect to the subset of $\{B\}, \{C\}, \{B, C\}$ are essentially the same, 2) the value function for B with respect to $\{C\}$ is 0, and 3) the value function for B with respect to $\{A, C\}$ is the same as that with respect to $\{A\}$:

$$E[f(\mathbf{x})|X_B] = E[f(\mathbf{x})|X_C] = E[f(\mathbf{x})|X_B, X_C]$$

$$\Delta_v(A, \{B\}) = \Delta_v(A, \{C\}) = \Delta_v(A, \{B, C\})$$

$$\Delta_v(B, \{C\}) = 0$$

$$\Delta_v(B, \{A, C\}) = \Delta_v(B, \{A\})$$

As a result, the Shapley values for A, B can be simplified to:

$$\phi_v(A) = \frac{1}{3}\Delta_v(A, \{\}) + \frac{2}{3}\Delta_v(A, \{B\})$$

$$\phi_v(B) = \frac{1}{3}\Delta_v(B, \{\}) + \frac{1}{6}\Delta_v(B, \{A\})$$

If we define a new model $f'(X_A, X_B) = f(X_A, X_B, X_B)$ which gives the same output for any data instance since $X_B = X_C$ and $f(\mathbf{x}) = f'(\mathbf{x})$. The game $v_{f,x}$ and $v_{f',x}$ are the same for all subsets of variables. Yet, if we limit the explanation to two features A, B instead of three, the Shapley values turn out to be different:

$$\phi'_v(A) = \frac{1}{2}\Delta_v(A, \{\}) + \frac{1}{2}\Delta_v(A, \{B\})$$

$$\phi'_v(B) = \frac{1}{2}\Delta_v(B, \{\}) + \frac{1}{2}\Delta_v(B, \{A\})$$

Notice the Shapley value $\phi'_v(B)$ for B in the two feature model is not equal to $\phi_v(B)$ nor $\phi_v(B) + \phi_v(C)$, the sum of Shapley values for the redundant features in the three features function. Hence, the relative feature importance of A and B depends on whether we consider C as a third feature, even though f and f' are essentially the same function (I Elizabeth Kumar et al. 2020).

Interventional Value Function

The interventional value function, $v_{f,x}(S) = E[f(X_S = x_S, X_{\bar{S}})]$, also has a mathematical issue because of the way it creates a joint distribution and introduces *out-of-distribution* samples (I Elizabeth Kumar et al. 2020).

For example, if a model f has three features X_1, X_2, X_3 such that $X_1, X_2 \sim N(0, 1)$ and $X_3 = X_1 X_2$, we want to find the value function for the addition of X_3 with respect to a known subset of features $S = \{X_1 = 1, X_2 = 2\}$, so we have to calculate the expected function output for $f(X_1 = 1, X_2 = 2, X_3)$. In the interventional value function, we calculate:

$$
\begin{aligned}
& E[f(X_1 = 1, X_2 = 2, X_3)] \\
&= \int P(X_1 = 1, X_2 = 2, X_3 = x_3) f(X_1 = 1, X_2 = 2, X_3 = x_3)\, dX_3 \\
&= \int P(X_1 = 1, X_2 = 2) P(X_3 = x_3) f(X_1 = 1, X_2 = 2, X_3 = x_3)\, dX_3
\end{aligned}
$$

In calculating the expectation, we are integrating over all possible values of x_3, but in reality $P(X_3 = x_3) = 0$ if $x_3 \neq x_1 x_2$, and the model f would have never seen such input combination in training. The model therefore has to extrapolate to make predictions on the out-of-distribution input, which defeats the purpose of explaining the working of the model for the task it is trained on. Compared to the conditional value function where we only integrate over the conditional distribution of f given values of known features, the interventional approach creates out-of-distribution examples that are not helpful in explaining the working of the model (I Elizabeth Kumar et al. 2020).

We illustrate in Fig 2.1 the differences between conditional and interventional estimation of the value function for a dataset. Suppose there is a distribution of all observed data where features X and Y are correlated Gaussian distributions. We draw samples from the data distribution to estimate $E[f(1, Y)]$ and $E[f(X, 2)]$ and to explain $f(1, 2)$ for some function f. Depending on whether the expectation is taken over $X|Y = 2$ and $Y|X = 1$ (left) or X and Y (right), we get all samples within the data distribution for the conditional approach, and all possible samples along $X = 1$ and $Y = 2$ for the interventional approach.

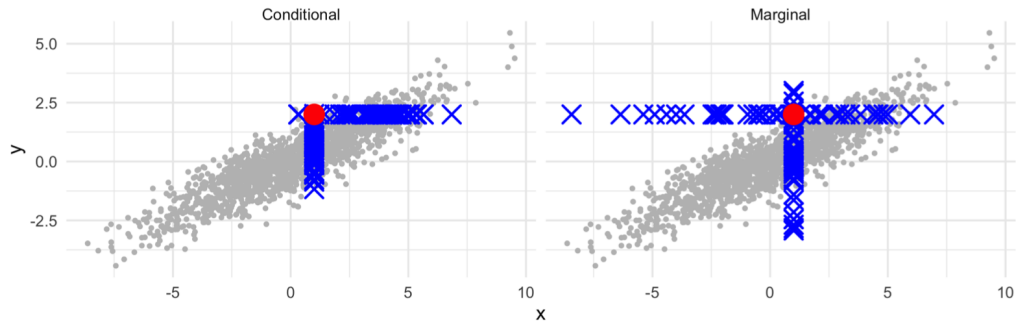

Fig. 2.1: Conditional and marginal samples drawn from the sample space to estimate $E[f(1,Y)]$ and $E[f(X,2)]$ to explain $f(1,2)$. (I Elizabeth Kumar et al. 2020)

2.5.2 Adversarial Vulnerabilities

We have shown that the interventional approach draws out-of-distribution samples as input to a machine learning model and estimate the value function. As a result, adversarial models can be devised to mask a biased classifier by detecting and giving unbiased output to out-of-sample data input, while returning the same biased output for real data and appearing more innocent by Shapley value interpretation.

For example, a credit-scoring model that illegally discriminates based on gender and races will have its output scores highly correlated with gender the race features in the dataset, and have high Shapley value for the two sensitive features. We want to construct an adversarial model that give the same biased output but has low Shapley values for the sensitive features when estimating with interventional value function.

Slack, Hilgard, Jia, et al. 2020 demonstrated that such adversarial model can be created to fool Shapley interpretations. Suppose a biased classifier $f(\mathbf{X})$ are trained and applied on real-world input data that follows the distribution $\mathcal{X}_{\text{dist}}$, and $\mathcal{X}$ is the distribution of the sample data that the adversarial model has access to. If given an input from the real-world distribution $\mathcal{X}_{\text{dist}}$, an adversarial classifier $e(\mathbf{X})$ will return the same biased prediction as $f(\mathbf{X})$. If detected an out-of-distribution input, the adversarial classifier $e(\mathbf{X})$ will return unbiased predictions based only

on features that are not correlated with sensitive features. Assume $\psi(\mathbf{X})$ is such unbiased classifier, the adversarial classifier e is:

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

To detect whether a data instance is in the distribution or not, we can simply construct a new dataset using available instances in $\mathcal{X}$. The new dataset contains original data X labelled with *out-of-sample* $= False$, and manipulated data instances X_p labelled with *out-of-sample* $= True$. We can generate each manipulated data point x_p by taking one $x \in X$ and replace some of its feature values to other values drawn from the marginal distribution. We then train a classifier on the dataset $X \cup X_p$ to detect out of sample data points.

The performance of the adversarial model in experiment is highly effective. The **COMPAS** dataset captures detailed information about the criminal history, jail and prison time, demographic attributes for 6172 defendants from Broward Couty, and the label is a measure of individual's risk scores. The sensitive feature is an attribute called "African-American." For the biased classifier f, the SHAP explanation always ranks the sensitive attribute first (Fig. 2.2). However, after the adversarial attack, the highest ranked feature is uncorrelated feature 1 most of the time, meaning its biases are successfully covered.

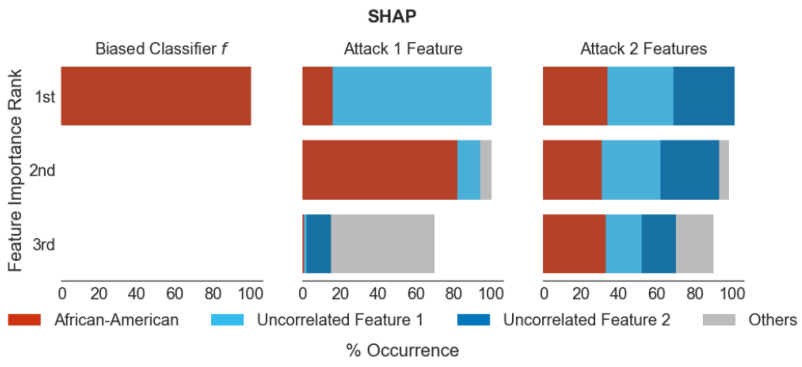

Fig. 2.2: Feature importance rank in SHAP for a biased classifier and the adversarial models (Slack, Hilgard, Jia, et al. 2020)

16

2.5.3 Human-centric Issues

Despite the aforementioned mathematical issues and the subsequent adversarial vulnerabilities, practitioners often use Shapley values as a tool for normative evaluation, such as deciding whether a model is acceptable or deployment-ready (I Elizabeth Kumar et al. 2020). The very purpose of calculating Shapley value is to bring better explanability to black box models, so the ultimate question we should ask about Shapley value is whether it helps us understand the model enough that people affected by the model can trust it and be treated fairly by it.

Disappointingly, Shapley values alone do not accomplish the goal as it does not give a full picture of how the game/model work. I. Kumar et al. 2020 gives a simple example that, if given two models f_1, f_1 with 3 input features x_1, x_2, x_3:

$$f_1 = x_1 + x_2 + x_3$$

$$f_2 = x_1 + 2x_2x_3$$

and a practitioner tries to explain the output $f_1(1, 1, 1) = 3$ and $f_2(1, 1, 1) = 3$. For both models, we end up with the Shapley value for x_1, x_2, x_3 equals 1 (assuming expected feature value is 0). However, increasing values of x_2 changes f_2 more than increasing the value of x_1, and the two functions are inherently different in their construct. This means that looking at Shapley values alone do not help us understand the working of the model completely, and there are information missing when we try to explain with Shapley values.

2.6 Quantifying limitations of Shapley Value - Shapley Residuals

How do we then quantify the missing information in the game not captured by Shapley value? I. Kumar et al. 2020 proposed the idea of Shapley residuals to capture information lost in Shapley values. They interpret Shapley values as the result of an orthogonal projection between vector spaces and quantify the residual lost as the kernel component of the projection.

First, we can set up the game as a function over the vertices of a N-dimensional hypercube, where each dimension takes a binary value representing the presence or absence of a feature, and the coordinates of each vertex on the hypercube represent a subset of players (Stern and Tettenhorst 2019).

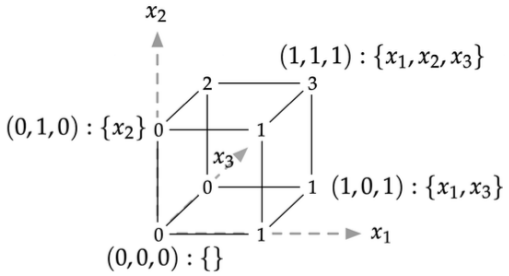

Fig. 2.3: Graphical representing of the game/model $f = x_1 + 2x_2x_3$ (I. Kumar et al. 2020)

As shown above, the game is represented as a hypercube $G(V, E)$ whose coordinates of the vertices represent a subset S of players, and the value of the function at each vertex $\mathbf{v} \in V$ is the expected model output $v(S)$ for that subset of players.

Now we can set up the edges of the hypercube, each of which connects two vertices representing two subsets of features in the game. The two vertices along an edge have all identical features, except the i^{th} feature for an edge that is parallel to the i axis, since the two end points of the edge have the i^{th} feature to be 0 and 1. Therefore, we can see the values along the edges as the delta of value function between S and $S' = S \cup \{i\}$. This can be more clearly illustrated in the figure below:

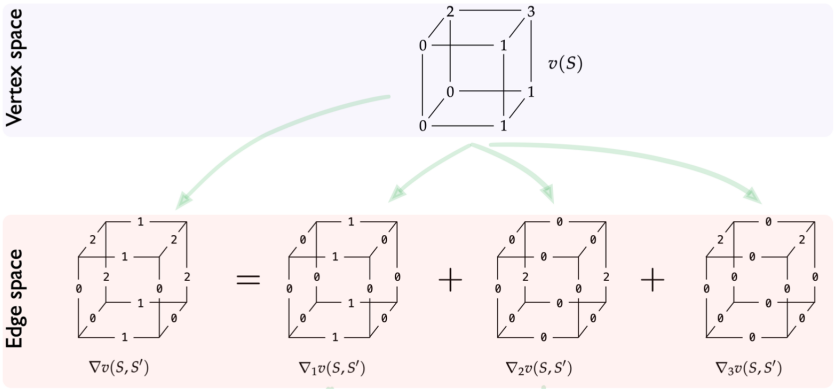

Fig. 2.4: Graphical representation of the delta of value function with respect to each feature in model $f = x_1 + 2x_2x_3$ (I. Kumar et al. 2020)

18

We can define the mapping between the value functions on the vertices to function on the edges by a discrete gradient operator $\nabla : \mathbb{R}^V \to \mathbb{R}^E$ as:

$$\nabla \mathbf{v}(S, S \cup \{i\}) = v(S \cup i) - v(S)$$

and a partial gradient ∇_i with respect to feature i:

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

Now suppose we have a game in which the player interactions are simple and additive. This means that every player adds a fixed value $v(i)$ to a coalition S independent of the composition of S. In this case, Shapley values can be easily used for feature importance measures since each feature's contribution is fixed and all features' contributions simply add up to the total model prediction. We call such a game inessential if for all S, $v(S) = \sum_{i \in S} v(i)$. We can express the inessentiality of a game by looking at the gradients on the hypercube:

Proposition 1 (I. Kumar et al. 2020). The game $\mathbf{v}$ *is inessential if and only if for each player* i*, there exists* $v_i \in \mathbb{R}^V$ *such that* $\nabla_i \mathbf{v} = \nabla \mathbf{v}_i$.

We can understand the equality $\nabla_i \mathbf{v} = \nabla \mathbf{v}_i$ by looking at an inessential game $f = x_1 + x_2 + x_3$ as shown in the figure below. $\nabla_i \mathbf{v}$ is the partial gradient for each feature i taken on game f we're explaining here (the same process shown for a different function in Fig 2.4). $\nabla \mathbf{v}_i$ represents the gradient taken on the cube for a game with only feature i. The three games for each feature sums up to the game f, and the gradient on the i^{th} game $\nabla \mathbf{v}_i$ is the same as the partial gradient $\nabla_i \mathbf{v}$ of the game f with respect to i.

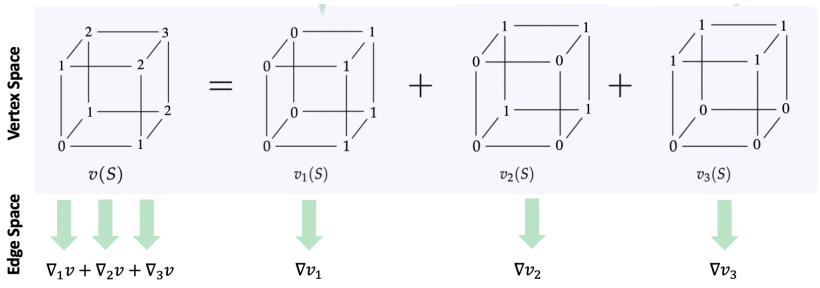

Fig. 2.5: Graphical representation of an inessential game defined by the function $f = x_1 + x_2 + x_3$

19

The equality may seem obvious for the above function, but a game may not be inessential in general and we cannot find $\mathbf{v}_i$ such that $\nabla_i \mathbf{v} = \nabla \mathbf{v}_i$. Instead, we can find the closest such $\mathbf{v}_i$ to minimize the least square error:

$$\nabla \mathbf{v}_i = \min_{\mathbf{x} \in \mathbb{R}^V, x(\emptyset)=0} ||\nabla \mathbf{x} - \nabla_i \mathbf{v}||$$

This means that we are effectively finding the closest inessential game $\mathbf{v}_i$ with respect to feature i. We can look back at the game defined by the function $f = x_1 + x_2 x_3$ which is not ineseential:

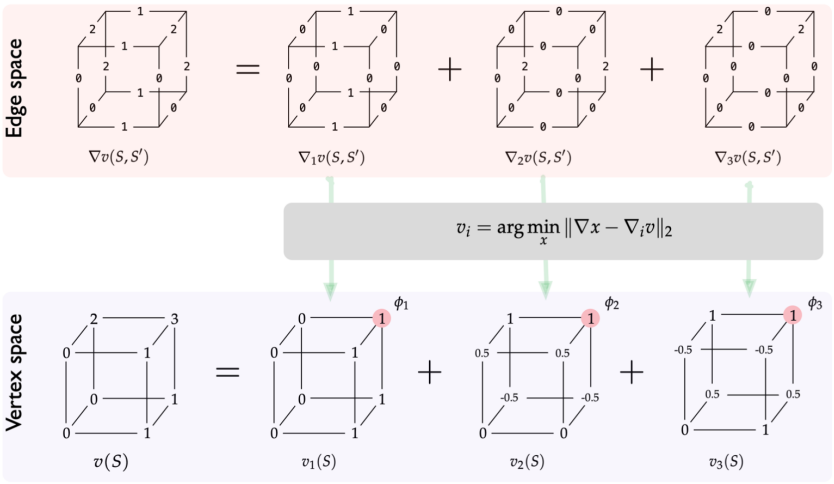

Fig. 2.6: Finding the least square error inessential game for $f = x_1 + x_2 x_3$. Notice that $\nabla \mathbf{v}_2 \neq \nabla_2 \mathbf{v}$ and $\nabla \mathbf{v}_3 \neq \nabla_3 \mathbf{v}$ since the game is not inessential and $v_2(S)$ and $v_3(S)$ are only the closest approximate inessential game (I. Kumar et al. 2020).

We can then quantify the degree of deviation of the game from inessentiality by looking at the difference between the partial gradient of our game and the inessential one.

$$r_i = \nabla_i \mathbf{v} - \nabla \mathbf{v}_i$$

where r_i is a vector with one value for each edge of the hypercube and it is a measure of deviation from inessentiality. This vector is zero if and only if the game is essential according to Proposition 1. We call r_i the Shapley Residual of player i.

Continuing with our earlier example for $f = x_1 + x_2 x_3$, once we found the closest inessential game $\mathbf{v}_i(S)$ with respect to feature i, we can calculate the difference between $\nabla_i \mathbf{v} - \nabla \mathbf{v}_i$ and find the residual:

20

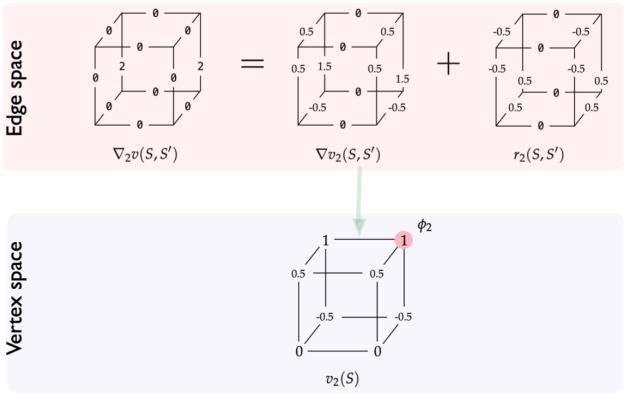

Fig. 2.7: Construction of Shapley residual r_2 for feature 2 using the best approximate inessential game v_2 we found. r_2 quantifies the deviation of $\nabla_2 \mathbf{v}$ from $\nabla \mathbf{v}_2$ (I. Kumar et al. 2020).

2.7 Quantifying Uncertainty of Shapley Value Estimation

While Shapley Residual provides a theoretical framework to quantify uncertainty due to the inessentiality of the model, in practice, we still need to first calculate the value function at each vertex of the hypercube (representing model output with a subset of players). When we calculate the value functions and subsequently the Shapley values with estimation techniques such as the one in section 2.3, it leads to uncertainty in the process.

In section 2.3, we described an estimation algorithm to calculate the Shapley value by sampling from a collection of data instances. The estimated Shapley value for a feature is its expected marginal contributions over all samples, which has an uncertainty due to the random process.

Merrick and Taly 2020 introduces a way to quantify this uncertainty. First, we define a single-reference game $v_{\mathbf{x},\mathbf{r}}$ that simulates feature absence by replacing the feature value with the value from a specific reference input $\mathbf{r}$.

$$v_{\mathbf{x},\mathbf{r}}(S) = f(\mathbf{z}(\mathbf{x}, \mathbf{r}, S)) - f(\mathbf{r})$$

21

where the function $\mathbf{z}(\mathbf{x}, \mathbf{r}, S)$ is called the composite input, which agrees with the input $\mathbf{x}$ on all features in S and with $\mathbf{r}$ on all features not in S. In Algorithm 1, the composite input is $\mathbf{x}_{+j} = (x_{(1)}, \ldots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$.

The attributions from a single-reference game explain the difference between the prediction for the input and the prediction for the reference. In Algorithm 1, the reference point is $\mathbf{x}_{-j} = (x_{(1)}, \ldots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$.

Merrick and Taly 2020 proves that the expected value of many single-reference games is an unbiased estimator of the true Shapley value depending on the type of distribution $\mathcal{D}^{ref}$ we draw references from (conditional or marginal):

$$\phi(v_{\mathbf{x}}, \mathcal{D}^{ref}) = \mathbb{E}_{\mathbf{R} \sim \mathcal{D}^{ref}}[\phi(v_{\mathbf{x},\mathbf{r}})]$$

With the above formulation, we can quantify attribution uncertainty by estimating the standard error of the mean (SEM) across a sample of Shapley values from single-reference games. In terms of the sample standard deviation (SSD), 95% CIs on the mean attribution ($\bar{\phi}$) from a sample of size M are given by (Merrick and Taly 2020):

$$\bar{\phi} \pm \frac{1.96 \cdot SSD(\{\phi(v_{\mathbf{x},\mathbf{r_i}})_{i=1}^{M}\})}{\sqrt{M}}$$

Merrick and Taly 2020 went on to derive a general SEM-based Confidence Interval that quantifies the combined uncertainty from the sampling of references, as described above, and sampling-based approximations of Shapley values.

Let us consider a generic estimator $\hat{\phi}_i^{(\mathbf{G})}(v_{\mathbf{x},\mathbf{r}})$ parameterized by some random sample $\mathbf{G}$. For example, in Algorithm 1, $\mathbf{G} = (\mathbf{O}_m)_{m=1}^{k}$ represents a random sample of feature orderings and let $pre_j(\mathbf{O})$ denotes the features up to feature of interest j:

$$\hat{\phi}_j^{(\mathbf{G})}(v_{\mathbf{x},\mathbf{r_j}}) = \frac{1}{M} \sum_{m=1}^{M} \Delta_v^m(x_j)$$

$$= \frac{1}{M} \sum_{m=1}^{M} v(pre_j(\mathbf{O}_m) \cup \{j\}) - v(pre_j(\mathbf{O}_m))$$

As we considered 1) the random sampling of data instances $\mathbf{R} \sim \mathcal{D}^{ref}$ to approximate the value functions and marginal contribution and 2) the random ordering $\mathbf{G}$ to approximate the different subsets/coalitions, we have taken into account the

22

estimation on both the weights and the marginal contribution when calculating the Shapley values as a weighted sum of marginal contributions:

$$\phi_j(v_{\mathbf{x}}, \mathcal{D}^{ref}) = \mathbb{E}_{\mathbf{R}}\mathbb{E}_{\mathbf{G}}[\hat{\phi}_j^{(\mathbf{G})}(v_{\mathbf{x},\mathbf{R}})]$$

Since $\mathbf{G}$ is independent of $\mathbf{R}$, this expectation can be Monte Carlo estimated using the sample mean of the sequence $(\hat{\phi}_j^{(\mathbf{g}_m)}(v_{\mathbf{x},\mathbf{r}_m}))_{m=1}^{M}$. Similarly, we can capture the uncertainty using standard error of the mean. A 95% CIs on the mean attribution ($\bar{\phi}$) from a sample of size M are given by (Merrick and Taly 2020):

$$\bar{\phi} \pm \frac{1.96 \cdot SSD((\hat{\phi}_j^{(\mathbf{g}_m)}(v_{\mathbf{x},\mathbf{r}_m}))_{m=1}^{M})}{\sqrt{M}}$$

2.8 Quantifying Uncertainty of SHAP - Bayesian SHAP

As we defined the theoretical limits of Shapley values using Shapley Residuals and quantified the sampling uncertainties in the estimation of Shapley value, we can investigate further into the uncertainty of SHAP model - a linear model using Shapley value explanation. Slack, Hilgard, Singh, et al. 2020 proposed a Bayesian SHAP model to accomplish that.

Recall from section 2.3 that SHAP is a linear explanation model. As defined earlier, g denotes the explanation model that we intend to learn to explain a function $f(\mathbf{x})$ at the instance $\mathbf{x}$. To approximate a linear explanation model around $\mathbf{x}$, we need to randomly draw samples around $\mathbf{x}$, a process called perturbation. We define $\mathcal{Z}$ as a set of n randomly sampled instances, and the proximity between any $z \in \mathcal{Z}$ and $\mathbf{x}$ is given by $\pi_x(z) \in \mathbb{R}$.

The original SHAP explanation model for every $z \in \mathcal{Z}$ can be written below using a vectorized form:

$$y = g(\mathbf{x}) = f(\mathbf{x}) = \phi^T z$$

where $\phi \in \mathbb{R}^N$ is a vector and ϕ_i is the Shapley value for feature i.

In our statistical analysis, there are two sources of uncertainties for the SHAP model: 1) the uncertainty associated with the values of the feature importance scores ϕ_i and, 2) uncertainty associated with how well g captures the local decision surface of the underlying black box model f (Slack, Hilgard, Singh, et al. 2020).

We capture this uncertainty by looking at SHAP output as a probability distribution conditioned on data samples z from perturbation , some prior belief of noise σ^2, and the Shapley values ϕ. We are adding Gaussian noise on the original linear model with zero bias and variance proportional to noise σ^2 and inversely proportional to the proximity function between $\mathbf{x}$ and z:

$$y|z, \phi, \sigma^2 \sim \phi^T z + \mathcal{N}(0, \frac{\sigma^2}{\pi_x(z)})$$

$$\sigma^2 \sim \text{Inv-}\chi^2(n_0, \sigma_0^2)$$

$$\phi|\sigma^2 \sim \mathcal{N}(\phi_0, \sigma^2\Sigma_0)$$

We assume the variance of the noise inversely proportional to the proximity function so we allow more room for error for points further from $\mathbf{x}$, while points close to $\mathbf{x}$ are modelled more accurately with a small variance. We assume prior beliefs that the noise σ^2 follows an inverse chi-square distribution with some initial small values of n_0 and σ_0^2, and the Shapley value ϕ follows a multivariate Gaussian distribution with some small mean ϕ_0 and variance-covariance matrix Σ_0, usually initialized with $Diag(1, ..., 1)$ so ϕ do not correlate and is sparse.

The first source of error we're interested in is the the uncertainty associated with ϕ, which is simply given by $\phi|\sigma^2 \sim \mathcal{N}(\phi_0, \sigma^2\Sigma_0)$. The second source of error, the uncertainty on explanation model g, is given by the noise term $\mathcal{N}(0, \frac{\sigma^2}{\pi_x(z)})$ and we call it ϵ.

We will find an expression for these two uncertainties by looking at the posterior distribution on ϕ and σ^2, which turns out to be normal and Inverse chi-square distributions respectively due to the corresponding conjugate priors we assumed earlier.

$$\sigma^2|\mathcal{Z}, Y \sim \text{Inv-}\chi^2(n, s^2)$$

$$\phi|\sigma^2, \mathcal{Z}, Y \sim \mathcal{N}(\hat{\phi}, V_\phi\sigma^2)$$

$$\hat{\phi} = V_\phi Z^T diag(\Pi_x(\mathcal{Z}))Y \; where \; V_\phi = (\mathcal{Z}^T diag(\Pi_x(\mathcal{Z}))\mathcal{Z} + I)^{-1}$$

$$s^2 = \frac{1}{n}[(Y - \mathcal{Z}\hat{\phi})^T diag(\Pi_x(\mathcal{Z}))(Y - \mathcal{Z}\hat{\phi}) + \hat{\phi}^T\hat{\phi}]$$

The derivation is shown in Slack, Hilgard, Singh, et al. 2020 but will be omitted here due to its complexity. So far, we only found the posterior distribution on ϕ and

not yet on ϵ. Hence, we will do another complicated derivation and it turns out that ϵ follows a Student's t distribution:

$$\epsilon|\mathcal{Z}, Y \sim t_{(\mathcal{V}=n)}(0, s^2)$$

With the probability density function derived for ϕ and ϵ, which are normal and t distribution respectively, we can calculate the confidence interval for ϕ and ϵ given any samples set $\mathcal{Z}$ we draw to estimate $\mathbf{x}$.

By studying the distributions, we notice that as the number of perturbations sampled around $\mathbf{x}$ goes to ∞: (1) the estimate of ϕ converges to the true importance scores, and its uncertainty converges to 0. (2) uncertainty of the error term ϵ converges to the bias of the local linear model g (Slack, Hilgard, Singh, et al. 2020).

However, it's unpractical to take an finite number of samples around $\mathbf{x}$ to achieve zero uncertainty. We want to sample enough data points so the uncertainty falls below a certain threshold. A naive algorithm will be randomly sampling points around $\mathbf{x}$ until the uncertainty is below the desired threshold. (Slack, Hilgard, Singh, et al. 2020) moved on to propose a more efficient batch-sampling method that strategically prioritizes perturbations whose predictions we are most uncertain about.

The uncertainty batch-sampling algorithm is given below in **Algorithm 1**. First, we randomly sample N instances (perturbations) around data point $\mathbf{x}$ and obtain a set of data $\mathcal{Q}$. Then, we generate a distribution $\mathcal{Q}_{\text{dist}}$ over N data points in $\mathcal{Q}$ by computing a probability $exp(z^T V_\phi z + 1)s^2 / \sum_{z \in \mathcal{Q}} exp(z^T V_\phi z + 1)s^2$ for each $z \in \mathcal{Q}$. We then draw B perturbations from $\mathcal{Q}_{\text{dist}}$ and input them into the blackbox for labels for B. Using the B instances as well as the N initial perturbations and their corresponding black box labels, we fit a local linear model g as discussed earlier. If the resulting explanation satisfies the desired certainty, terminate early and return the local explanation, otherwise, repeat all steps above (Slack, Hilgard, Singh, et al. 2020).

Algorithm 2 Uncertainty sampling for local explanations (Slack, Hilgard, Singh, et al. 2020)

Require: Perturbation size S, Preliminary perturbation size N, Batch size B, Model f, Data instance X, Explanation Model g with Predictive Variance ϕ_g, Candidate perturbation batch size A

procedure UNCERTAINTY SAMPLE
  Initialize data set D and add N initial perturbations, $(\mathcal{Z}, f(\mathcal{Z}))$.
  Fit g on D
  for $i = 1$ to $S - N$ **do**
    if i mod. $B = 0$ **then**
      Generate set of candidate perturbations $\mathcal{Q}$ of size A
      Draw B perturbations into $\mathcal{Q}_{\text{new}}$ from $\mathcal{Q}_{\text{dist}} \sim \frac{exp(\phi_g(\mathcal{Q}))_{j \in |\mathcal{Q}|}}{\sum exp(\mathcal{Q})}$
      $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathcal{Q}_{\text{new}}, f(\mathcal{Q}_{\text{new}}))$; Fit g on D
    end if
  end for
  return g
end procedure

26

3

Conclusion

We have discussed the definition of Shapley value and its application in measuring feature importance and explaining black box model. We recognized the mathematical, adversarial, and human-centric issues with Shapley values. We discovered that the limitation and uncertainties of Shapley value are actually three-fold: the inessentiality of the model leads to missing information about the composition of the representing game and feature interactions; the Mean-Squared-Errors in samplings of the coalition distributions and a feature's marginal contributions in the respective coalition lead to uncertainty in Shapley value estimation; the non-linearity of the data surface in constructing local linear model leads to uncertainty in SHAP model output. Respectively and separately, we found three methods to quantify the uncertainty. We can use Shapley Residuals to quantify the game's deviation from inessentialness; we can use the Sample Standard Deviation of the Monte-Carlo samples to calculate the uncertainty in Shapley value estimation; we can use Bayesian SHAP to quantify the uncertainty in SHAP model and uses uncertainty sampling technique to efficiently reduce the uncertainty.

27

4

Statement of The Problem

Although we have identified three sources of uncertainty related to Shapley value and respective methods to quantify the uncertainties, it is unclear how different types of uncertainty can be related to one another and how much each uncertainty measurement technique can inform practitioners in real-life applications.

We want to create artificial datasets that: 1) separately highlight one source of uncertainty and mainly captured by one uncertainty measurement method. For example, we want to create models with highly correlated features so there's a large Shapley Residuals. We can create data with sparse data points so estimation through sampling has large standard error. We can create data with highly non-linear surfaces so SHAP explanation models will have large variances. 2) emphasize two or more sources of uncertainties and demonstrate the relations between uncertainty measures. For example, data with highly correlated features, which would have larger Shapley residual theoretically, can also lead to higher uncertainties measured by Bayesian SHAP, presumably due to higher variance in the perturbed dataset. 3) have all types of uncertainties to test the overall trust-worthiness of Shapley value explanations and examine when the combined effect of three types of uncertainty is large enough to render Shapley value explanation unreliable. For example, we can use hypothesis testing techniques or an overall confidence interval to quantify the quality of Shapley explanation.

We can also investigate and compare techniques for reducing uncertainty and measure their effectiveness on different types of uncertainties. One potential method is a sampling technique I saw in Benati et al. 2019, which uses orthogonal projection to improve Shapley value estimation results towards the symmetry and efficiency properties, which at the same time also reduces the Mean-Standard-Error. We can compare that to the uncertainty sampling technique in Bayesian SHAP and evaluate how well they reduce different types of uncertainty.

Last but not the least, we can apply uncertainty measurement techniques in real-life dataset and examine their performance and usefulness. In particular, we want to explain existing models' prediction on the outcome of chemical experiments, which consists of high dimensional correlated data. We want to investigate how different types of uncertainty measurements inform Chemists about the model's behaviors and their Shapley explanations. We want to study how Chemists can interpret the Shapley values and their uncertainties to advance their understanding of chemistry.

29

References

Belle, Vaishak and Ioannis Papantonis (2020). “Principles and Practice of Explainable Machine Learning”. In: *arXiv preprint arXiv:2009.11698* (cit. on pp. 4, 5).

Benati, Stefano, Fernando López-Blázquez, and Justo Puerto (2019). “A stochastic approach to approximate values in cooperative games”. In: *European Journal of Operational Research* 279.1, pp. 93–106 (cit. on p. 28).

Kumar, I Elizabeth, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler (2020). “Problems with Shapley-value-based explanations as feature importance measures”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)* (cit. on pp. 5, 9, 12–15, 17).

Kumar, IE, C Scheidegger, S Venkatasubramanian, and S Friedler (2020). “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. 5, 17–21).

Lundberg, Scott M and Su-In Lee (2017). “A unified approach to interpreting model predictions”. In: *Advances in neural information processing systems*, pp. 4765–4774 (cit. on pp. 4, 11, 12).

Merrick, Luke and Ankur Taly (2020). “The Explanation Game: Explaining Machine Learning Models Using Shapley Values”. In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, pp. 17–38 (cit. on pp. 6, 21–23).

Molnar, Christoph (2019). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. `https://christophm.github.io/interpretable-ml-book/` (cit. on pp. 2, 3, 7, 10).

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. 2).

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). “Model-agnostic interpretability of machine learning”. In: *ICML Workshop on Human Interpretability in Machine Learning (WHI)* (cit. on p. 4).

Slack, Dylan, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju (2020). “Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods”. In: *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES)* (cit. on pp. 5, 15, 16).

Slack, Dylan, Sophie Hilgard, Sameer Singh, and Himabindu Lakkaraju (2020). “How Much Should I Trust You? Modeling Uncertainty of Black Box Explanations”. In: *arXiv preprint arXiv:2008.05030* (cit. on pp. 6, 23–26).

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 p. 18).