이 학 박 사 학 위 논 문

Towards interpretable machine learning : A methodology for screening interactions in functional ANOVA model

해석 가능한 머신러닝 모형을 위한 functional ANOVA model의 교호작용 스크리닝 방법론

2023년 8월

서울대학교 대학원

통계학과

최 용 찬

Towards interpretable machine learning :
A methodology for screening interactions in
functional ANOVA model
해석 가능한 머신러닝 모형을 위한
functional ANOVA model의 교호작용 스크리닝 방법론

지도교수 김 용 대

이 논문을 이학박사 학위논문으로 제출함
2023년 4월

서울대학교 대학원
통계학과
최 용 찬

최용찬의 이학박사 학위논문을 인준함
2023년 6월

위 원 장	오 희 석	(인)
부위원장	김 용 대	(인)
위 원	Junyong Park	(인)
위 원	신 예 은	(인)
위 원	김 동 하	(인)

Towards interpretable machine learning : A methodology for screening interactions in functional ANOVA model

By

Yongchan Choi

A Thesis
Submitted in fulfillment of the requirement
for the degree of
Doctor of Philosophy
in Statistics

Department of Statistics
College of Natural Sciences
Seoul National University
August, 2023

ABSTRACT

Towards interpretable machine learning : A methodology for screening interactions in functional ANOVA model

Yongchan Choi
The Department of Statistics
The Graduate School
Seoul National University

In this thesis, we propose a post-process interpretation method. Recently, machine learning has received great attention due to its remarkable predictive accuracy in various fields. Despite their strong predictive performance, machine learning models have usually lack of interpretability since this improvement in predictive performance has been achieved through increased model complexity. This makes it difficult for people to understand the models.

So, We develop a new post-process interpretation method called *Meta-ANOVA*, which interprets a given predictive model. To interpret the model, we devise a framework for detecting interactions

in the model. Moreover, we propose an efficient learning algorithm for the functional ANOVA model when the interactions are given.

Keywords: Explainable AI, Model interpretation, Interaction detection, Functional ANOVA model, Identifiability

Student Number: 2016 – 20277

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are two things to be considered when evaluating predictive models : prediction accuracy and interpretability. Over the recent decades, many predictive models with high prediction ability, such as ensemble-based models and deep neural networks, have been developed, and they have received much attention [Chouiekh and Haj, 2018; Devlin et al., 2018; He et al., 2016; Radford et al., 2019; Shen et al., 2017].

Despite their strong predictive performance, machine learning models have usually lack of interpretability since this improvement in predictive performance has been achieved through increased model complexity. This makes it difficult for people to understand the models. That is, they are treated as black-box. Black-box models could be acceptable to low-risk tasks. However, in high-risk tasks such as cancer diagnosis and self-driving cars system, models that cannot be interpreted are hard to be used. The need

for trustworthiness of modern machine learning models in real-world applications has led to the revival of the field of eXplainable Artificial Intelligence (XAI).

In this thesis, we introduce a novel method called *Meta-ANOVA*, which aims to construct an interpretable model for a given predictive model. The key idea of our method is to approximate the given predictive model using a functional ANOVA model. To approximate the model efficiently

1. We propose an interaction detection method.

2. We introduce an efficient learning algorithm for functional ANOVA model.

Meta-ANOVA efficiently learns the functional ANOVA model without losing the prediction power of a given model and it is theoretically well-grounded. To conduct the first step successfully, we also provide an efficient interaction detection algorithm similar to Apriori algorithm.

In Chapter 2, we briefly review XAI methods and the functional ANOVA model. In Chapter 3, we explain the proposed method. In Chapter 4, we present the results of numerical study.

Chapter 2

Review

2.1 Review : Post-process interpretation methods

XAI methods can be classified into several perspectives. We first categorize the methods into two groups. One is transparent model design and the other is post-process interpretation.

Transparent model design aims to build models that can be trained to interpret their predictions (white-box models). These models provide simultaneous prediction and interpretation, making them reliable for their interpretability. Linear model and decision trees are typical transparent models. However, creating a white-box model often requires introducing constraints, which can lead to performance degradation.

Post-process interpretation is a method used to understand the inference process of a pre-trained model. The prediction is ob-

tained from the pre-trained model, and post-process methods are then applied to gain interpretation or insight into the prediction. Unlike transparent model design, post-process interpretation does not lead to performance degradation, but the interpretability of the results is often less reliable. Ensuring interpretation reliability becomes a key concern for these methods. Post-process interpretation can be further divided into two categories: local and global interpretation. Global interpretation involves understanding how a given model makes predictions across the entire input space. On the other hand, local interpretation focuses only on a specific input vector, rather than considering the entire input space.

2.1.1 Global interpretation

Tsang et al. [2017] devise a global interpretation method, called *NID* (Neural Interaction Detection). *NID* first introduces a framework for detecting statistical interactions from a pre-trained neural network. *NID* utilizes the structure of the neural network. In order for the network to have a specific interaction, there should be at least one hidden node with strong weight connections to corresponding input nodes. The Figure 2.1 shows the weight connections between the first and third input nodes and the first hidden node of the first hidden layer. Let $W^{(l)} \in \mathbf{R}^{p_l \times p_{l-1}}$ and $\mathbf{b}^{(l)}$ $(l = 1, \ldots, L)$ are weight matrices and bias vectors where p_l is the number of hidden units in the l-th layer. The i-th row and j-th column of W^l are denoted by $W^{(l)}_{i,:}$ and $W^{(l)}_{:,j}$.

NID measures the weight connections of interaction $\mathcal{I}$ for each

node of the first hidden layer as follows : $\mu\left(\left|W_{i,\mathcal{I}}^{(1)}\right|\right)$ where $\mu(\cdot)$ is averaging function. $\mu(\cdot) = \min(\cdot)$ is used in practice . After measuring the weight connections of all hidden nodes in the first layer, the influence of the hidden nodes on the output is given as follows :

$$\mathbf{z}^{(\ell)} = \left|\mathbf{w}^{y}\right|^{\top} \left|\mathbf{W}^{(L)}\right| \cdot \left|\mathbf{W}^{(L-1)}\right| \cdots \left|\mathbf{W}^{(\ell+1)}\right|.$$

NID defines the interaction strength $\omega_i(\mathcal{I})$ of an interaction $\mathcal{I}$ for i-th hidden unit as

$$\omega_i(\mathcal{I}) = z_i^{(1)} \mu\left(\left|\mathbf{W}_{i,\mathcal{I}}^{(1)}\right|\right).$$

Finally, the importance score of the neural network $\omega(\mathcal{I})$ is defined by

$$\omega(\mathcal{I}) = \sum_{j=1}^{p_1} \omega_i(\mathcal{I}).$$

NID provides importance scores for the interactions that the trained model has. However, *NID* is applicable only to neural networks.

2.1.2 Local interpretation

The additive Feature Attribution method is a framework for local interpretation. Let f and $\mathbf{x}$ be a given model and input vector respectively. The objective is to interpret f at $\mathbf{x}$ using an explanation model $g_{\mathbf{x}}$. The explanation model $g_{\mathbf{x}}$ is a linear function of binary variables :

$$g_{\mathbf{x}}\left(\mathbf{z}_{\mathbf{x}}\right) = \phi_{\mathbf{x},0} + \sum_{j=1}^{M} \phi_{\mathbf{x},j} z_{\mathbf{x},j}$$

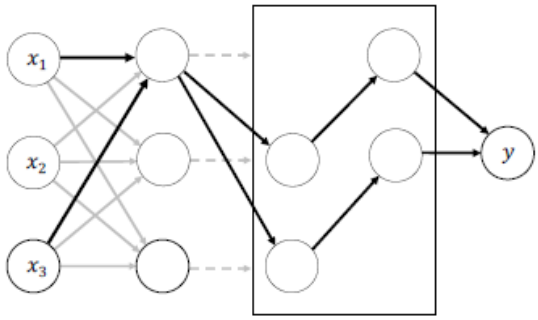

Figure 2.1: An illustration of an interaction within a fully connected neural network. The bold black arrow means strong weight connection. The top node of the first hidden layer takes inputs from $(1, 3)$ with strong connection.

where $\mathbf{z_x} \in \{0,1\}^M$ is a simplified binary input vector for $\mathbf{x}$ and M is the number of simplified input features. Simplified input refers to transforming the original input data into a human-understandable format or representation. For instance, consider an image of a dog where the values of each pixel in the image are entirely incomprehensible to humans. However, humans can understand and interpret whether the image contains the dog's nose or not. This illustrates that simplified input can be defined differently based on the domain of the input variables. Assuming that the transformation to simplified input is feasible.

Additive Feature Attribution method tries to ensure $g_{\mathbf{x}}(\mathbf{z_x}) \approx f(\mathbf{x})$. Then, it interprets the original model f at $\mathbf{x}$ using $g_{\mathbf{x}}$. There are two representative methods to estimate $\phi_{\mathbf{x},j}$. One is *LIME*

[Ribeiro et al., 2016] and the other is *SHAP-values* [Lundberg and Lee, 2017].

[Ribeiro et al., 2016] utilizes local linear regression using Gaussian kernel where . Let $\pi_{\mathbf{x}}(\mathbf{x}') = \exp\left(-\|\mathbf{x}' - \mathbf{x}\|^2)/\sigma^2\right)$ be a Gaussian kernel. The objective function of *LIME* is given as :

$$\mathcal{L}\left(f, g_{\mathbf{x}}, \pi_x\right) = \sum_{x' \in \mathcal{X}} \pi_{\mathbf{x}}(\mathbf{x}') \left(f(\mathbf{x}') - g_{\mathbf{x}}\left(\mathbf{z}'\right)\right)^2 + \Omega(g_{\mathbf{x}})$$

where $\mathcal{X}$ is input dataset, $\mathbf{z}'$ is a simplified input of $\mathbf{x}'$ and Ω is L_1-penalty. [Ribeiro et al., 2016] minimizes the objective function and interprets f at $\mathbf{x}$ via $g_{\mathbf{x}}$.

[Lundberg and Lee, 2017] uses Shapley value estimation method [Shapley et al., 1953] to estimate $g_{\mathbf{x}}$. The estimate of $\phi_{\mathbf{x},j}$ is

$$\phi_{\mathbf{x},j} = \sum_{\tilde{\mathbf{x}}' \subseteq x'} \frac{|\tilde{\mathbf{x}}'|!\,(M - |\tilde{\mathbf{x}}'| - 1)!}{M!} \left[f\left(\tilde{\mathbf{x}}'\right) - f\left(\tilde{\mathbf{x}}' \backslash j\right)\right]$$

where $|\tilde{\mathbf{x}}'|$ is the number of non-zero entries in $\tilde{\mathbf{x}}'$, and $\tilde{\mathbf{x}}' \subseteq x'$ represents all $\tilde{\mathbf{x}}'$ vectors where the non-zero entries are a subset of the non-zero entries in $\mathbf{x}'$. They shows that *SHAP-values* is the unique estimation method of additive feature attribution that satisfies Local accuracy, Missingness and Consistency properties. See [Lundberg and Lee, 2017] for details of the properties.

2.2 Review : Learning algorithms of functional ANOVA model

The functional ANOVA model aims to decompose the model into a sum of components representing main effects and interactions. It allows us to understand how each component for mains and interactions contributes output of the model. Thus, it is transparent box model discussed in 2.1.

The functional ANOVA model f has the following form :

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{p} f_j(x_j) + \sum_{j<k} f_{j,k}(x_j, x_k) + \dots \tag{2.1}$$

where f_js are the main components and $f_{j,k}$s are second-order interaction components and so on. The identifiability for the functional ANOVA model assured by the averaging operator [Gu, 2013]. Let $\mathcal{X}_j$ be the support for X_j and μ_j be the probability measure on $\mathcal{X}_j$. The support for $\mathbf{X} = (X_1, \dots, X_p)$ is denoted as $\mathcal{X} = \prod_{j=1}^{p} \mathcal{X}_j$. The averaging operator A_j on f is defined as

$$A_j f = \int_{\mathcal{X}_j} f d\mu_j.$$

Condition . The functional ANOVA model f holds identifiability condition if $f_{\mathbf{j}}$ satisfies $A_j f_{\mathbf{j}} = 0$ for all $\mathbf{j} \subset \{1, 2, \dots, p\}$ and $j \in \mathbf{j}$.

There are several learning algorithms of the functional ANOVA model [Friedman, 1991; Gu, 2013; Kim et al., 2005; Lin and Zhang, 2006].

2.2.1 Smoothing Spline-ANOVA [Gu, 2013]

SS-ANOVA (Smoothing Spline-ANOVA) is the most representative functional ANOVA model. It is assumed that $f \in \mathcal{F}$ where $\mathcal{F}$ is a RKHS(Reproducing Kernel Hilbert Space). In general, each component is in the second-order Sobolev Hilbert space. The lth-order Sobolev Hilbert Space $\mathcal{S}_l$ is defined as

$$\mathcal{S}_l = \left\{ g : g, g', \ldots, g^{(l-1)} \text{are absolutely continuous}, g^{(l)} \in \mathcal{L}_2 \right\}$$

For computational issue, main and second-order interactions are considered. Denote the norm in the RKHS $\mathcal{F}$ by $\|\cdot\|$. *SS-ANOVA* finds $f \in \mathcal{F}$ to minimize :

$$\frac{1}{n}\sum_{i=1}^{n}\left\{y_i - f\left(x_i\right)\right\}^2 + \lambda\sum_{\alpha=1}^{d}\theta_{\alpha}^{-1}\left\|P^{\alpha}f\right\|^2$$

where d is the number of components in f, P^{α} is the orthogonal projection of f onto $\mathcal{F}^{\alpha}$ and $\theta_{\alpha} \geq 0$. In *SS-ANOVA*, the identifiability condition is ensured due to orthogonality.

2.2.2 *COSSO* [Lin and Zhang, 2006]

COSSO (COmponent Selection and Smoothing Operator) is a learning algorithm for a functional ANOVA model that enables component selection. *COSSO* uses the sum of RKHS norms as the penalty, not the squared sum of the norm. *COSSO* finds $f \in \mathcal{F}$ to minimize :

$$\frac{1}{n}\sum_{i=1}^{n}\left\{y_i - f\left(x_i\right)\right\}^2 + \tau_n\sum_{\alpha=1}^{d}\left\|P^{\alpha}f\right\|$$

Note that the penalty of *COSSO* is not a norm in $\mathcal{F}$. However, it is convex. [Lin and Zhang, 2006] shows the existence of *COSSO* estimate using the convexity and asymptotic properties of *COSSO*. Moreover, they propose an iterative algorithm to solve the objective function of *COSSO* more easily. See [Lin and Zhang, 2006] for details of the properties and algorithm.

2.2.3 *MARS* [Friedman, 1991]

MARS (Multivariate adaptive regression spline) is another learning algorithm of functional ANOVA model. Each component consists of the modification of the CART. *MARS* utilizes a pair of hinge functions. We denote the hinge function as $\phi_c(x) = \max(0, x - c)$.

MARS uses a stepwise forward-backward procedure for building functional ANOVA models. First, *MARS* adds a pair of hinges $\{\phi_c(x), -\phi_c(x)\}$ and finds the coefficients for each hinge function. To select components, *MARS* finds the knot c and selects the coefficients that satisfy the identifiability condition (forward procedure). After adding each hinge pair and coefficients, *MARS* prunes the existing components to prevent overfitting (backward procedure). Pruning is conducted based on GCV (Generalized Cross-Validation). This forward–backward procedure is repeated until a pre-defined stopping rule is satisfied.

2.2.4 *ANOVA-Boosting* [Kim et al., 2005]

ANOVA-Boosting is a boosting model on the functional ANOVA model. For computational issue, *ANOVA-Boosting* only considers main and 2nd-order interaction. *ANOVA-Boosting* model f is

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{p} f_j(x_j) + \sum_{j<k} f_{jk}(x_j, x_k)$$

where f_j and f_{jk} are estimated by a linear combination of base learners. Let $\mathcal{G}_j$ be the set of decision trees with two terminal node split by the variable X_j. For $g_j \in \mathcal{G}_j$, we can formularize g_j as :

$$g_j(x_j) = \theta_L I(x_j \leq s_j) + \theta_R I(x_j > s_j).$$

By identifiability condition(i.e. $A_j g_j = 0$), we have

$$\theta_L P(X_j \leq s_j) + \theta_R P(X_j > s_j) = 0.$$

This means that one of θ_L and θ_R uniquely defines the other one. Let $\mathcal{G}_{jk}$ be the set of decision trees with four terminal node split by the variables X_j and X_k. For $g_{jk} \in \mathcal{G}_{jk}$, we can formularize g_{jk} as

$$\begin{aligned} g_{jk}(x_j, x_k) = \theta_{LL} I(x_j \leq s_j, x_k \leq s_k) + \theta_{LR} I(x_j \leq s_j, x_k > s_k) \\ + \theta_{RL} I(x_j > s_j, x_k \leq s_k) + \theta_{LL} I(x_j > s_j, x_k > s_k) \end{aligned}$$

By the identifiability condition, $A_j g_{jk} = 0$ and $A_k g_{jk} = 0$ should be hold.One of $\theta_{LL}, \theta_{LR}, \theta_{RL}$ and θ_{RR} uniquely defines the others same as main case. Similar manner, identifiable base learner for higher order interaction can be defined.

For each step, *ANOVA-Boosting* finds the identifiable base learner that explains the residual most and aggregates it.

Algorithm 1: ANOVA-boosting algorithm for regression problem.

Require: $\{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, \ldots, n\}$: Training dataset.

Require: M : The number of trees, η : The learning rate

Require: ℓ : The squared error loss.

Require: $f_0(\mathbf{x}) = \bar{y}$.

for $m = 1, \ldots, M$ **do**

- Compute the residual $r^{(i)} = y^{(i)} - f_{m-1}(\mathbf{x}^{(i)})$ for $i = 1, \ldots, n$
- Fit base learners $g_j \in \mathcal{G}_j$ for $j \in J_1$ and $g_{\mathbf{j}}$ for $\mathbf{j} \in J_2$ to the targets $r^{(i)}$
- $\mathbf{j}^* = \arg\min_{\mathbf{j} \in (J_1 \cup J_2)} \sum_{i=1}^{n} \ell(r^{(i)}, g_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}^{(i)}))$
- $f_m = f_{m-1} + \eta g_{\mathbf{j}^*}$

end

Result: f_M

Chapter 3

Proposed Method

3.1 Introduction

We present the proposed *Meta-ANOVA* framework to interpret any given predictive model. The key idea of our method is to approximate a given predictive model to functional ANOVA model , which can be interpreted both locally and globally simultaneously. The main difference between global and local interpretation model is the scope of interpretation. Local interpretation models can explain a block-box model only near a single datum. To understand the model globally, the methods sholud be applied to entire training dataset. It is computationally expensive, and even there is no guarantee that this can make the user to understand model globally. On the other hand, our method can interpret a given model for any input since it approximates a given model for the entire input space.

To improve global approximation accuracy, we propose a method to detect statistical interactions that a given predictive model has learned.

Let $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$ be an p-dimensional input vector and $f : \mathcal{X} \to \mathbb{R}$ be a given predictive model. Let $[p] = \{1, \ldots, p\}$ and $\mathbf{j} \subset [p]$ denote feature index set and interaction respectively. Let $\mathbf{x}_\mathbf{j} = (x_j, j \in \mathbf{j})$ be a sub-input vector and $J_k = \{\mathbf{j} \subset [p] : |\mathbf{j}| = k\}$ is k-order interaction set. We denote $\mathbf{j}^c = [p] \setminus \mathbf{j}$.

Before explaining our method, we introduce the definition of statistical interactions. We follow the definition of statistical interactions provided [Sorokina et al., 2008].

Definition. Function f does not have an interaction of (i, j) if it can be expressed as the sum of two functions $f_{\setminus i}$ and $f_{\setminus j}$ where $f_{\setminus i}$ does not depend on x_i and $f_{\setminus j}$ does not depend on x_j :

$$f(\mathbf{x}) = f_{\setminus i}(\mathbf{x}_{[p]\setminus i}) + f_{\setminus j}(\mathbf{x}_{[p]\setminus j})$$

This is the definition of second-order interactions for the function f. Higher-order interactions are defined similarly. From the definition, kth-order interaction can only exist if all its corresponding $(k-1)$th-order interactions exist. For example, the interaction $(1, 2, 3)$ can only exist if the interactions $(1, 2), (1, 3), (2, 3)$ should exist. We use this property in Section 3.2

3.2 Step 1 : Search for interactions

We introduce a method to detect significant interactions from a black-box model. This step is main contribution of our method. This is because most post-process interpretation methods do not consider interactions, and transparent box models like the functional ANOVA model primarily focus only on main and second-order interactions. However, our method is an efficient way to find higher-order interactions. We first consider the case of continuous input variables and then extend to general input variables.

3.2.1 Continuous input variables

Assume all input variables are continuous and f is differentiable. For a given model f, we consider the following functional decomposition:

$$f(\mathbf{x}) = \beta_0 + \sum_{k=1}^{p} \sum_{\mathbf{j} \in J_k} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}})$$

For identifiability, we assume the following condition :

Condition 1 (Identifiability condition). For any $\mathbf{j}$, let $f_{\mathbf{j}+}(\mathbf{x}) = \sum_{\mathbf{j}' > \mathbf{j}} f_{\mathbf{j}'}(\mathbf{x}_{\mathbf{j}'})$. If $f_{\mathbf{j}+}(\mathbf{x})$ does not depend on $\mathbf{x}_{\mathbf{j}^c}$, then $f_{\mathbf{j}'} \equiv 0$ for all $\mathbf{j}' > \mathbf{j}$.

Note that for any functional decomposition $f(\mathbf{x}) = \beta_0 + \sum_{k=1}^{p} \left\{ \sum_{\mathbf{j} \in J_k} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}) \right\}$, we can redefine β_0 and $f_{\mathbf{j}}$s such that f satisfies the Condition. An example of the Condition is as follows :

- Assume $f(\mathbf{x}) = \beta_0 + f_1(x_1) + f_2(x_2) + f_{1,2}(x_{1,2})$ where $f_1(x_1) = x_1$, $f_2(x_2) = x_2$, $f_{1,2}(x_{1,2}) = 3$.
- $f_{1+}(\mathbf{x}) = f_{1,2}(x_{1,2})$ does not depend on x_2.
- Redefine β_0 and $f_{\mathbf{j}}$s such that

$$f(\mathbf{x}) = (\beta_0 - 3) + f_1(x_1) + f_2(x_2)$$

Before explaining the main theorem used to find interactions, we first define the partial derivative. Let $\mathcal{X}_j \subset \mathbb{R}$ be the domain of x_j for each $j \in [p]$. Also, let $\mathcal{X} = \prod_{j=1}^{p} \mathcal{X}_j$ and $\mathcal{X}_{\mathbf{j}^c} = \prod_{j \in \mathbf{j}^c} \mathcal{X}_j$. For a given $j \in [p]$ and a function $f : \mathcal{X} \to \mathbb{R}$, the partial derivative of f at $\mathbf{x}$ w.r.t. the index j given as

$$D_j f(\mathbf{x}) := \lim_{\epsilon \to 0} \frac{f(\mathbf{x} + \epsilon \mathbf{e}_j) - f(\mathbf{x})}{\epsilon}$$

where $\mathbf{e}_j$ is a p-dimensional vector whose j-th element is 1 and 0 otherwise. Consider a set $\mathbf{j} = (j_1, ..., j_k)$. The partial derivative of f at $\mathbf{x}$ w.r.t. the index set $\mathbf{j}$ is given as

$$D_{\mathbf{j}} f(\mathbf{x}) := D_{j_1} \circ \cdots \circ D_{j_k} f(\mathbf{x}).$$

Theorem 3.2.1. *For given* $\mathbf{j}$, $f_{\mathbf{j}'} = 0$ *for all* $\mathbf{j}' > \mathbf{j}$ *(*$\mathbf{j}' > \mathbf{j}$ *means* $\mathbf{j}' \supsetneq \mathbf{j}$*) if and only if*

$$\mathbb{E}_{\mathbf{X}'_{\mathbf{j}} \sim Q_{\mathbf{j}}} \left[\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} \right] = 0, \tag{3.1}$$

$Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$ are any distributions defined on the same support for $P_{\mathbf{j}}$ and $P_{\mathbf{j}^c}$ where $P_{\mathbf{j}}$ and $P_{\mathbf{j}^c}$ are the marginal distributions of $\mathbf{X}_{\mathbf{j}}$ and $\mathbf{X}_{\mathbf{j}^c}$.

The proof of the Theorem is in Appendix A.1. Here is an example of the Theorem.

- Assume $f(\mathbf{x}) = \beta_0 + f_1(x_1) + f_2(x_2) + f_{1,2}(x_1, x_2)$ where $\mathbf{x} = (x_1, x_2) \in \mathbf{R}^2$, $\beta_0 = 1$, $f_1(x_1) = x_1$, $f_2(x_2) = x_2$ and $f_{1,2}(x_1, x_2) = \theta x_1 x_2$. Let $\mathbf{j} = \{1\}$
- We can get the partial derivative $D_{\mathbf{j}} f(\mathbf{x}) = 1 + \theta x_2$.
- From the partial derivative, $\mathbb{E}_{\mathbf{X}'_{\mathbf{j}} \sim Q_{\mathbf{j}}} \left[\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} \right] = \theta^2 \mathrm{Var}(X'_2)$.
- If $\theta \neq 0$, then $\mathbb{E}_{\mathbf{X}'_{\mathbf{j}} \sim Q_{\mathbf{j}}} \left[\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} \right] > 0$.
- If $\theta = 0$ (i.e. $f_{1,2} = 0$), then $\mathbb{E}_{\mathbf{X}'_{\mathbf{j}} \sim Q_{\mathbf{j}}} \left[\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} \right] = 0$.

We call $\mathbb{E}_{\mathbf{X}'_{\mathbf{j}} \sim Q_{\mathbf{j}}} \left[\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} \right]$ a interaction importance score for $\mathbf{j}$ and denote it as $I(\mathbf{j}; f)$(abbr. $I(\mathbf{j})$). Note that $I(\mathbf{j})$ is not the importance score for $\mathbf{j}$ itself but the importance score for the higher order interactions including $\mathbf{j}$.

Note that $Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$ are any distributions defined on the same support for $P_{\mathbf{j}}$ and $P_{\mathbf{j}^c}$. This implies that if we have the model and only know the support of each input variable (i.e., the possible values it can take), without having any training data, we can compute the statistics.

3.2.2 General input variables

In Section 3.2.1, we assume that all input variables are continuous and f is differentiable. In this case, the partial derivative can be

utilized. However, the partial derivative cannot be used when

1. The predictive model is non-differentiable(e.g. tree-based model)

2. A categorical variable is used as input variable.

In these cases, we cannot use the statistics we proposed. To overcome this issue, we replace the differentiation with the difference.

Assume X_js are ordinal variable. Let $\mathcal{X}_j = \{x_{j1}, \cdots, x_{jn_j}\}$ be the domain of X_j for $j \in [p]$ where $x_{j1} < \cdots < x_{jn_j}$ and n_j is the number of values that X_j can have. For a given $j \in [p]$ and a function $f : \mathcal{X} \to \mathbb{R}$, we define the partial ordinal derivative of $f(\mathbf{x})$ w.r.t. the index j given as

$$d_j f(\mathbf{x}) := \begin{cases} f(\mathbf{x} : x_j = x_{ji}) - f(\mathbf{x} : x_j = x_{ji-1}) & \text{if } i = n_j, \\ f(\mathbf{x} : x_j = x_{ji+1}) - f(\mathbf{x} : x_j = x_{ji}) & \text{if } i = 1, \\ \left(f(\mathbf{x} : x_j = x_{ji+1}) - f(\mathbf{x} : x_j = x_{ji-1})\right)/2 & \text{otherwise.} \end{cases}$$

Consider a vector $\mathbf{j} = (j_1, ..., j_k)$ and a function $f : \mathcal{X} \to \mathbb{R}$. We generalize the partial ordinal derivative w.r.t. the index set $\mathbf{j}$ like partial derivative as follows :

$$d_{\mathbf{j}} f(\mathbf{x}) := d_{j_1} \circ \cdots \circ d_{j_k} f(\mathbf{x}).$$

$d_{\mathbf{j}}$ can be used when X_js are ordinal variables. Moreover, it can be utilized when X_j is categorical input variable. This is because categorical input variable is transformed to dummy variable(or one-hot encoded variable) in general. Details about categorical input are summerized in Appendix A.2.

For continuous variable, we can expect that Theorem 3.2.1 approximately holds for $d_{\mathbf{j}}$ when enough samples are given. Based on theorem and partial ordinal derivative, we redefine the score of the higher order interactions of $\mathbf{j}$ as follows:

$$I(\mathbf{j}) = \mathbb{E}_{\mathbf{X}'_{\mathbf{j}} \sim Q_{\mathbf{j}}} \left[\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ d_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} \right]$$

Note that $I(\mathbf{j})$ can be applied to the non-differentiable model.

Based on the interaction importance score $I(\mathbf{j})$, we suggest the interaction detection algorithm. We use the property of the statistical interactions discussed in Section 3.1 :

kth-order interaction can only exist if all its corresponding $(k-1)$th-order interactions exist.

From the property, we can detect significant interactions sequentially as Apriori algorithm used for searching association rules. The algorithm is described in Algorithm 2. When kth-order interaction $\mathbf{j}$ is given, its $(k-1)$th-order interactions are denoted by $\mathcal{R}(\mathbf{j}) = \{\mathbf{j} \setminus j : j \in \mathbf{j}\}$.

When using Algorithm 2, there are two practical issues to consider. One is too many $\mathbf{j}$ with $I(\mathbf{j}) > 0$ and the other is the choice of $Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$.

3.2.3 Threshold selection

Highly-complex model may have almost all interactions even if they have litter effect on the output. Thus, we use a threshold to

Algorithm 2: Interaction searching algorithm.

Require: K: maximum degree for interaction

Initialization: $k = 1$

Initialization: $S_1 = [p]$

while $k \leq K$ **do**

Calculate the followings:

- $k \leftarrow k + 1$
- $C_k = \{\mathbf{j} \in S_{k-1} : I(\mathbf{j}) > 0\}$.
- $S_k = \{\mathbf{j}_1 \cup \mathbf{j}_2 : \mathbf{j}_1, \mathbf{j}_2 \in C_k, |\mathbf{j}_1 \cup \mathbf{j}_2| = k, \mathcal{R}(\mathbf{j}_1 \cup \mathbf{j}_2) \subset C_k\}$.

end

Result: $S = \cup_{k=1}^{K} S_k$

exclude minor interactions. The meaning of $I(\mathbf{j})$ is the importance score for all $\mathbf{j}'$ where $\mathbf{j}' > \mathbf{j}$. From the meaning, we adopt an approach of excluding interactions that have significantly lower importance scores compared to the maximum importance score. For screening kth-order interactions, let $I^k = \{I(\mathbf{j}) : \mathbf{j} \in S_{k-1}\}$. Then, $\tau_k = (\gamma \times \max I^k)$ for pre-defined hyper-paremeter γ where $0 < \gamma < 1$. Instead of using 0 as threshold, we use τ_k as the threshold for each k step.

3.2.4 Choice of $Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$

To calculate $I(\mathbf{j})$, we need to select $Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$. where $Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$ are distributions of $\mathbf{X}'_{\mathbf{j}}$ and $\mathbf{X}'_{\mathbf{j}^c}$ defined on the same support for $P_{\mathbf{j}}$ and $P_{\mathbf{j}^c}$. Note that Theorem 3.2.1 holds for any distributions

$Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$. Thus, we can consider $\mathbf{X}'_{\mathbf{j}}$ and $\mathbf{X}'_{\mathbf{j}^c}$ are independent. Moreover, it is preferable for $Q_{\mathbf{j}}$ and $Q_{\mathbf{j}^c}$ to be closer to the true distributions $P_{\mathbf{j}}$ and $P_{\mathbf{j}^c}$. Therefore, we use the distributions $Q_{\mathbf{j}} = \hat{P}_{\mathbf{j}}$ and $Q_{\mathbf{j}^c} = \hat{P}_{\mathbf{j}^c}$ where $\hat{P}_{\mathbf{j}}$ and $\hat{P}_{\mathbf{j}^c}$ are the marginal empirical distributions of $X_{\mathbf{j}}$ and $X_{\mathbf{j}^c}$.

3.3 Step 2 : Learning approximation model

Let S be the set of interactions obtained in step 1. The functional ANOVA model with interactions $S = \cup_{k=1}^{K} S_k$ is defined by

$$f^a(\mathbf{x}) = \beta_0 + \sum_{\mathbf{j} \in S_k} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}). \tag{3.2}$$

The goal of step 2 is to learn f^a to approximate the predictive model f as much as possible. We can formularize as following:

$$\hat{f}^a = \underset{f^a}{\operatorname{argmin}} \, E_P \ell \left(f(X), f^a(X) \right)$$

where P is a distribution of input variables. We use a Boosting model to approximate the original model. However, boosting models are inherently non-interpretable. To address this, we employ *ANOVA-boosting* [Kim et al., 2009], which is a boosting model based on the functional ANOVA model, making it interpretable. To efficiently utilize the interactions obtained in step 1, we introduce two modifications to *ANOVA-boosting*. We refer to this modified version as Modified *ANOVA-boosting*.

3.3.1 Modified *ANOVA-boosting*

For the first modification, we introduce a greedy algorithm to iteratively select the split value for each base learner. *ANOVA-boosting* uses all possible search to find split value for each base learner. Thus, the computational complexity is $\mathcal{O}(n^k)$ for kth-order interaction. Note that the advantage of our method is that it effectively detects the higher-order interactions. If we use an all possible search, the computations for higher order interactions are almost infeasible, even for third-order interactions.

On the other hand, the computational complexity of the greedy algorithm increases linearly with k. Thus, we use a greedy algorithm similar to the decision tree. Here is the example of greedy search for Modified *ANOVA-boosting*. The objective is to find a base learner for interaction $(1, 2)$.

1. Train base learners $g_1 \in \mathcal{G}_1$ and $g_2 \in \mathcal{G}_2$.

2. Choose the best one g_j among g_1 and g_2 where

$$j = \underset{k \in \{1,2\}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(r^{(i)}, g_k(x_k^{(i)})).$$

3. (Assume $j = 1$) For a given s_1 where s_1 is split value for g_1, find best split value s_2 for $g_{1,2}$.

By applying the greedy algorithm, modified *ANOVA-boosting* can efficiently consider higher-order interactions obtained in step 1. However, it is well known that greedy algorithm may cause the

local minima problem(e.g. XOR problem). To overcome this issue, we introduce second modification.

We applies the bumping method(Hastie et al. [2009]) to modified *ANOVA-boosting*. This technique uses bootstrap sampling to avoid the local minima problem. To find a base learner for interaction $\mathbf{j}$, draw B bootstrap dataset $\mathbf{Z}^1, \ldots, \mathbf{Z}^B$ and train B base learners using $\mathbf{Z}^b (b = 1, \ldots, B)$. Lastly choose the best one.

3.3.2 Synthetic dataset

We train f^a using the modified *ANOVA-boosting* with synthetic dataset. The reason for using synthetic dataset is to fill in the support of the input data in order to better approximate the origin model. We use two types of synthetic dataset : One is interpolation samples and the other is extrapolation samples. Let $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are two training input samples. We generate interpolation and extrapolations samples as follows :

1) Interpolations samples

$$\tilde{\mathbf{x}} = \lambda \mathbf{x}^{(i)} + (1 - \lambda)\mathbf{x}^{(j)}$$

where $\lambda \sim Beta(\alpha, \alpha)$ for $\alpha > 0$.

2) Extrapolation samples

$$\tilde{\mathbf{x}} = -\delta \mathbf{x}^{(i)} + (1 + \delta)\mathbf{x}^{(j)}$$

where $\delta \sim U(0, \beta)$ for $\beta > 0$. Thus, training dataset for *Meta-ANOVA* consists of two datasets :

- Original training dataset $T = \{(\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)})) : i = 1, \ldots, n\}$

- Synthetic dataset $N = \{(\tilde{\mathbf{x}}^{(i)}, f(\tilde{\mathbf{x}}^{(i)})) : i = 1, \ldots, n_s\}$ where n_s is pre-defined the number of synthetic samples.

Using modified *ANOVA-boosting* and synthetic dataset, we train f^a to approximate f. *Meta-ANOVA* algorithm is described in Algorithm 3

3.3.3 Interpreting black-box model via *Meta-ANOVA*

Let $\hat{f}^a$ be the estimated *Meta-ANOVA* model. Then, it can be represented as follows :

$$\hat{f}^a(\mathbf{x}) = \beta_0 + \sum_{k=1}^{K} \sum_{\mathbf{j} \in S_k} \hat{f}_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}).$$

Let $\mathbf{x}$ be a given input datum. Then, our model can give the predicted value $\hat{f}_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}})$ for each component. Thus, it can provide local interpretation. To interpret the black-box model globally, we introduce the importance score of each component as following:

$$IS(\mathbf{j}) = \mathrm{Var}\left(\hat{f}_{\mathbf{j}}(\mathbf{X}_{\mathbf{j}})\right)$$

Under the identifiability condition for functional ANOVA model, the larger the interactions importance score, the greater the influence on the prediction.

Algorithm 3: *Meta-ANOVA* algorithm

Require: $T \cup N = \{(\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)})) : i = 1, \ldots, n + n_s\}$:
Training and synthetic dataset.

Require: $S = \cup_{k=1}^{K} S_k$: The detected interactions

Require: M : The number of trees, $\eta = 1$: learning rate.

Require: $f_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{n} \ell(f(\mathbf{x}^{(i)}), \gamma)$

for $m = 1, \ldots, M$ **do**

- Compute the residual $r^{(i)} = f(\mathbf{x}^{(i)}) - f_{m-1}(\mathbf{x}^{(i)})$ for $i = 1, \ldots, n + n_s$
- $r_i = -\left[\frac{\partial L(y_i, a)}{\partial a}\right]_{a=f_{m-1}(\mathbf{x}_i)}$

  for $\mathbf{j} \in S$ **do**

  - Draw B bootstrap datasets $\mathbf{Z}^1, \ldots, \mathbf{Z}^B$.
  - Fit base learners $g_{\mathbf{j}}^b$, $b = 1, \ldots, B$ using greedy algorithm.
  - $\hat{g}_{\mathbf{j}} = \underset{g_{\mathbf{j}} \in \{g_{\mathbf{j}}^b : b=1,\ldots,B\}}{\operatorname{argmin}} \sum_{i=1}^{n+n_s} \ell(r^{(i)}, g_{\mathbf{j}}(\mathbf{x}^{(i)}))$

  end

- $\mathbf{j}^* = \underset{\mathbf{j} \in S}{\operatorname{argmin}} \sum_{i=1}^{n+n_s} \ell(r^{(i)}, \hat{g}_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}^{(i)}))$
- $f_m = f_{m-1} + \eta \hat{g}_{\mathbf{j}^*}$

end

Result: f_M

Chapter 4

Experiments

We conduct experiments to evaluate the performance of our method, focusing on two aspects : 1) How effectively our interaction search algorithm can find the true interactions, and 2) How well *Meta-ANOVA* approximates the given predictive model. The evaluation is carried out in two scenarios: 1) Simulated data, and 2) Real data.

4.1 Simulated data

Since ground-truth labels for interactions are not available in a general predictive model, we utilize ten synthetic functions that consist of second and higher-order interactions. All ten functions and input distributions are the same as described in Liu et al. [2020]; Tsang et al. [2017]. The list of synthetic function is in Table 4.1. We regard these functions as given predictive models and apply our framework.

Table 4.1: List of 10 synthetic functions used in the Simulated data experiments.

F_1	$\pi^{x_1 x_2}\sqrt{2x_3} - \sin^{-1}(x_4) + \log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7$
F_2	$\pi^{x_1 x_2}\sqrt{2\lvert x_3\rvert} - \sin^{-1}(0.5x_4) + \log(\lvert x_3 + x_5\rvert + 1) + \frac{x_9}{1+\lvert x_{10}\rvert}\sqrt{\frac{x_7}{1+\lvert x_8\rvert}} - x_2 x_7$
F_3	$\exp\lvert x_1 - x_2\rvert + \lvert x_2 x_3\rvert - x_3^{2\lvert x_4\rvert} + \log(x_4^2 + x_5^2 + x_7^2 + x_8^2) + x_9 + \frac{1}{1+x_{10}^2}$
F_4	$\exp\lvert x_1 - x_2\rvert + \lvert x_2 x_3\rvert - x_3^{2\lvert x_4\rvert} + (x_1 x_4)^2 + \log(x_4^2 + x_5^2 + x_7^2 + x_8^2) + x_9 + \frac{1}{1+x_{10}^2}$
F_5	$\frac{1}{1+x_1^2+x_2^2+x_3^2} + \sqrt{\exp(x_4 + x_5)} + \lvert x_6 + x_7\rvert + x_8 x_9 x_{10}$
F_6	$\exp(\lvert x_1 x_2\rvert + 1) - \exp(\lvert x_3 + x_4\rvert + 1) + \cos(x_5 + x_6 - x_8) + \sqrt{x_8^2 + x_9^2 + x_{10}^2}$
F_7	$(\arctan(x_1) + \arctan(x_2))^2 + \max(x_3 x_4 + x_6, 0) - \frac{1}{1+(x_4 x_5 x_6 x_7 x_8)^2} + \left(\frac{\lvert x_7\rvert}{1+\lvert x_9\rvert}\right)^5 + \sum_{i=1}^{10} x_i$
F_8	$x_1 x_2 + 2^{x_3+x_5+x_6} + 2^{x_3+x_4+x_5+x_7} + \sin(x_7 \sin(x_8 + x_9)) + \arccos(0.9x_{10})$
F_9	$\tanh(x_1 x_2 + x_3 x_4)\sqrt{\lvert x_5\rvert} + \exp(x_5 + x_6) + \log((x_6 x_7 x_8)^2 + 1) + x_9 x_{10} + \frac{1}{1+\lvert x_{10}\rvert}$
F_{10}	$\sinh(x_1 + x_2) + \arccos(\tanh(x_3 + x_5 + x_7)) + \cos(x_4 + x_5) + \sec(x_7 x_9)$

As discussed in Section 3.2, our method can detect interactions that model has learned. To evaluate our framework, We conduct experiments on synthetic functions $F_1 - F_{10}$. First, interaction detection is conducted (step 1). For synthetic functions, we select the threshold hyper-parameter $\gamma = 0$ because the object is to find true interactions. To calculate $I(\mathbf{j})$, we draw 100 samples from $Q_{\mathbf{j}}$ and 400 samples from $Q_{\mathbf{j}^c}$. We limit the maximum interaction order K to 4. The reason is that interpreting interactions beyond the 4th- order interaction becomes challenging.

After step 1, we approximate the synthetic functions utilizing interactions detected from step 1. The hyper-parameters for

modified *ANOVA-Boosting* are fixed by

$$(\eta, M, B, \alpha, \beta, n_s) = (0.1, 3000, 5, 0.5, 0.2, 5 \times n)$$

.

First, we assess the ability of our method to detect second-order interactions for synthetic functions. For a given second-order interaction $\mathbf{j} \in S_2$, we can estimate $IS(\mathbf{j}) = \text{Var}\left(\hat{f}_{\mathbf{j}}(\mathbf{X}_{\mathbf{j}})\right)$, the importance score of $\mathbf{j}$, using training dataset. We set $IS(\mathbf{j}) = 0$ for $\mathbf{j} \notin S_2$.

We compare our method to other existing methods that detect interactions including *Two-Way ANOVA* Fisher [1992], *HierLasso* (Hierarchical Lasso) Bien et al. [2013], *RuleFit* Friedman and Popescu [2008], *AG* (Additive groves Sorokina et al. [2008], *NID* (Neural Interaction Detection) Tsang et al. [2017] and *PID* (Persistence Interaction Detection) Liu et al. [2020]. *PID* is a similar method to *NID*. Like *NID*, *PID* is also a methodology for detecting interactions from a neural network. The main difference lies in the way it measures the importance of interactions. *PID* quantifies the strength of interactions based on the theory of persistent homology.

There is a notable distinction between *Meta-ANOVA* and other baseline methods. Our method is capable of detecting interactions for any predictive model, making it model-agnostic. On the other hand, other methods can only be applied to specific models. This implies that even when the true function is available, other methods would still need to train a model within their specific function

class to detect interactions. Additionally, if one wants to detect interactions from a particular model, other methods may not be applicable or may require re-training a model using their specific function class.

The AUC score of interaction strength is used as a performance comparison measure. We conducted 10 trials and removed the two trials with the highest and lowest AUC scores. The averaged AUC score results are presented in Table 4.2. It is evident that *Meta-ANOVA* can perfectly detect second-order ground-truth interactions.

Table 4.2: The results of synthetic functions : The comparison between *Meta-ANOVA* and other methods for AUCs of second order interactions

	ANOVA	*HierLasso*	*RuleFit*	*AG*	*NID*	*PID*	*Meta-ANOVA*
F_1	0.992	1.000	0.754	1.000	0.985	0.986	1.000
F_2	0.468	0.636	0.698	0.880	0.776	0.804	1.000
F_3	0.657	0.556	0.815	1.000	1.000	1.000	1.000
F_4	0.563	0.634	0.689	0.999	0.916	0.935	1.000
F_5	0.544	0.625	0.797	0.670	0.997	1.000	1.000
F_6	0.780	0.730	0.811	0.640	0.999	1.000	1.000
F_7	0.726	0.571	0.666	0.810	0.880	0.888	1.000
F_8	0.929	0.958	0.946	0.937	1.000	1.000	1.000
F_9	0.783	0.681	0.584	0.808	0.968	0.972	1.000
F_{10}	0.765	0.583	0.876	1.000	0.989	0.987	1.000
Average	0.721	0.698	0.764	0.870	0.951	0.957	1.000

Furthermore, *Meta-ANOVA* screening algorithm successfully detects all true interactions in synthetic functions. This means that all ground-truth interactions are included in the set S, even the highest order interactions.

Additionally, we use various values of γ to select threshold for mains and compare the average of importance scores $IS(\mathbf{j})$ for the detected 2nd-order interactions. The results are shown in Figure 4.1. These results suggest that the higher $I(\mathbf{j})$, introduced by our proposed method, the more likely our algorithm captures important interactions.

4.2 Real data

For Real data, we analyze five datasets : Calhousing Pace and Barry [1997], letter recognition Frey and Slate [1991], german credit Grömping [2019], Satellite and Online news dataset. Details about each dataset are summarized in Table 4.3. We split the data into train/validation/test sets in a ratio of 70/10/20 for all datasets.

For Real data, our method needs a pre-trained model. We consider following models:

1. Neural network with for hidden layers(140-100-60-20).
2. XGBoost with depth $3, 4, 5$.
3. Random forest

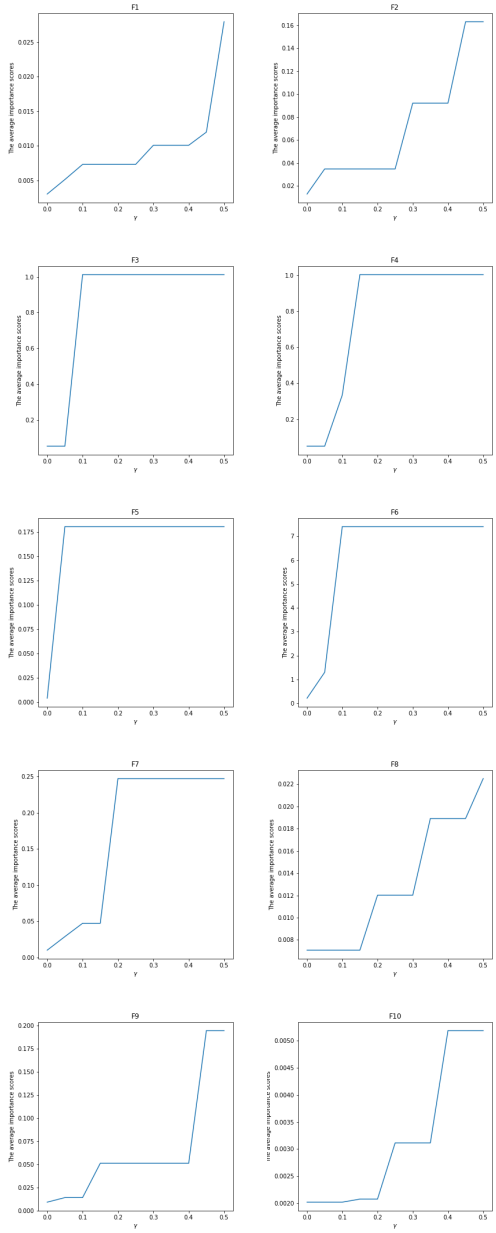

Figure 4.1: The results for synthetic function : The average of the importance scores for various γ.

Table 4.3: The descriptions of Real data.

Dataset	Number of samples	Number of features	Problem
Calhousing	21k	8	Regression
Letter-recognition	20k	16	Classification
German-credit	1k	61	Classification
Satellite	6k	36	Classification
Online news	40k	58	Classification

Table 4.4: Baseline model and its test error for Real data.

Dataset	Model	Test error
Calhousing	NN	0.2063
Letter-recognition	NN	0.0255
German-credit	XGB4	0.1900
Satellite	XGB4	0.0668
Online news	XGB3	0.3316

The hyper-parameters for each model are selected using the validation set. We choose the model with the best test error (MSE for regression and misclassification rate for classification) for each dataset, and we refer to it as the baseline model. The results are summarized in Table 4.4. We can observe that the best test error models for each dataset are NN (Neural Network) and XGB (XGBoost), which are not interpretable due to their high complexity.

In the case of Real data, the true interactions of the baseline model are unknown. Therefore, we evaluate how well our method

Table 4.5: The results of *Meta-ANOVA* for Real data : The test error comparison between baseline model and *Meta-ANOVA*.

	Baseline model	*Meta-ANOVA*
Calhousing	0.2063	0.2188
Letter	0.0255	0.0481
German credit	0.1900	0.1900
Satellite	0.0668	0.0772
Online news	0.3316	0.3351

approximates the baseline model by comparing the test error of *Meta-ANOVA* to the baseline model. All the hyper-parameters required for *Meta-ANOVA*, except for γ, are kept the same. As mentioned earlier, complex models often include interactions that have little impact on the output. To exclude such interactions, we introduce the hyaper-parameter γ. In all experiments with Real data, we set $\gamma = 0.1$.

Table 4.5 shows the test error for the baseline model and *Meta-ANOVA*. From the results, we can confirm that our method shows almost the same test error compared to the baseline models, which means *Meta-ANOVA* approximates the baseline model effectively.

We check $IS(\mathbf{j})$, the importance score for interaction $\mathbf{j}$. Figure 4.3 shows the top 10 importance score for Calhousing dataset. It is well known that 'median income(7)' is closely related to house price. Moreover, 'Longitude' and 'latitude' (0,1) shows strength interactions. Through this results, we are able to confirm that the

baseline model considers 'median income' and location information as important features.

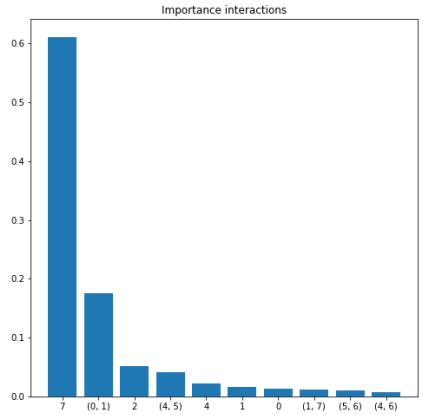

Figure 4.2: The top 10 importance scores for Calhousing dataset

4.3 Ablation studies

In this section, we report the results of ablation studies. The things to be examined in the ablation studies are as follows:

- Comparison with other learning algorithm of functional ANOVA model
- The sensitive analysis of γ
- The effect of bumping method
- The effect of synthetic dataset

4.3.1 Comparison with other learning algorithm of functional ANOVA model

If training data is available, we can use functional ANOVA models to train a transparent box model. This is because functional ANOVA models are inherently interpretable on their own. Therefore, we compare the performance of models when using functional ANOVA models directly and our method. To confirm this, we use two types of data : 1) Simulated data and 2) Real data.

In Simulated data, we compare our method to other functional ANOVA models when true function consists of main and second-order interactions. We consider the following true regression function [Lin and Zhang, 2006] :

$$h(\mathbf{x}) = g_1(x_1)+g_2(x_2)+g_3(x_3)+g_4(x_4)+g_1(x_3x_4)+g_2(\frac{x_1+x_3}{2})+g_3(x_1x_2)$$

where $g_1(t) = t$, $g_2(t) = (2t-1)^2$, $g_3(t) = \frac{\sin(2\pi t)}{2-\sin(2\pi t)}$ and $g_4(t) = 0.1\sin(2\pi t)+0.2\cos(2\pi t)+0.3\sin^2(2\pi t)+0.4\cos^3(2\pi t)+0.5\sin^3(2\pi t)$. We consider 10 input variables $X_1, \ldots, X_{10} \sim Unif(0,1)$. Note that h only uses four input variables. Therefore, $X_5, \ldots, X_{10}$ are uninformative. We consider two simulation cases.

- Case 1 : $n = 400$
- Case 2 : $n = 10,000$

We generate $y = h(x)+\epsilon$ where $\epsilon \sim N(0, 0.2546^2)$. ISE (Integrated squared error) is used as comparison measure same as [Lin and Zhang, 2006].

$$ISE = E\left(\hat{h}(X) - h(X)\right)^2$$

Table 4.6: The results of Simulated data : The comparison between *Meta-ANOVA* and other functional ANOVA algorithms.

Model	ISE(case 1)	ISE(case 2)
SS-ANOVA [Gu, 2013]	0.0617	**0.0117**
COSSO [Lin and Zhang, 2006]	**0.0431**	X
MARS2 [Friedman, 1991]	0.0483	0.0174
NN	0.0814	0.0121
Meta-ANOVA(NN)	0.0891	0.0132
Meta-ANOVA(*MARS2*)	0.0517	0.0188

We estimate ISE by Monte Carlo integration using 10,000 test points. Neural network with for hidden layers(100-100-50-20) is used as predictive model for our method. We use the main and second-order interactions for *SS-ANOVA*, *COSSO* and *MARS*. We conduct experiments for various values of hyper-parameters and select the best model for each algorithm.

For our method, we draw 100 samples from $Q_{\mathbf{j}}$ and 400 samples from $Q_{\mathbf{j}^c}$ to calculate $I(\mathbf{j})$. We set $\gamma = 0.1$ and $(M, B, \alpha, \beta, n_s) = (3000, 5, 0.5, 0.2, 5 \times n)$. The results are in Table 4.6.

MARS2 denotes *MARS* with depth 2. X means computationally infeasible. In case 1, NN shows bad performance. This is likely due to overfitting, which appears to occur because of the small sample size. Thus, our method using NN also shows bad performance. However, our method is model-agnostic. Thus, we test our method using *MARS*. The result shows that our method

using *MARS* has performance similar to *MARS*, and even better than *SS-ANOVA*.

In case 2, *SS-ANOVA* shows best ISE. However, our method for NN is pretty comparable with *SS-ANOVA* and outperforms *COSSO* and *MARS2*. Moreover, our method for *MARS* shows similar ISE compared to *MARS2*. Note that the true regression function consists of main and second interactions, which is a favorable experimental setting for the functional ANOVA model.

We conduct the comparison using Real data : Calhousing dataset. In Real data, we cannot use ISE because we don't know true function. Thus, we select MSE as comparison measure.

Table 4.7: The results of Real data : The comparison between *Meta-ANOVA* and other functional ANOVA algorithm.

Model	Test error
Meta-ANOVA(NN)	0.2188
SS-ANOVA	0.2484
COSSO	X
MARS4	0.2784

Table 4.7 shows the results for each functional ANOVA learning algorithm. Note that our method shows best test error compared to other algorithms. This is likely because there exist higher order interactions among the input variables to explain the response variable in the Calhousing dataset. Therefore, our method is expected to perform well on data where a significant number of

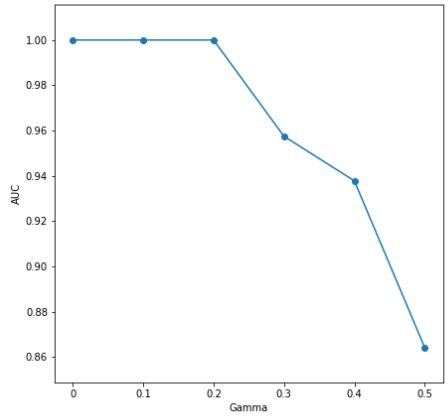

Figure 4.3: The results of sensitive analysis of γ using synthetic function : For each γ, we report the AUC scores of importance score.

higher order interactions are present.

4.3.2 Sensitive analysis of γ

In step 1, γ controls the number of interactions. We check how γ affects the results for *Meta-ANOVA*. We consider two cases : 1) Simulated data and 2) Real data. For Simulated data, we use F_5 in Table 4.1.

$$F_5(\mathbf{x}) = \frac{1}{1 + x_1^2 + x_2^2 + x_3^2} + \sqrt{\exp(x_4 + x_5)} + |x_6 + x_7| + x_8 x_9 x_{10}$$

.

The true interactions in F_5 are

- Second order : $(1, 2), (1, 3), (2, 3), (4, 5), (6, 7), (8, 9), (8, 10), (9, 10)$

- Third order : $(1, 2, 3), (8, 9, 10)$

We regard F_5 as a given predictive model and apply *Meta-ANOVA*. The hyper-paremters for our method is same as other experiments except for γ. We conduct experiments for various γ and measure the AUC score of the importance score for each γ. The Figure 4.3 suggests that our method shows robust results on Simulated data as long as γ is not too large.

We also conduct sensitive analysis of γ using Real data. Similar to previous experiments, we don't know the true interactions of the predictive model. Thus, we use test errors as comparison measure. The results are in Figure 4.4. The results show that our method does not sensitive to γ as long as it is not too large.

4.3.3 The effect of bumping method

In step 2, we introduce bumping method to *ANOVA-Boosting* since greedy algorithm can cause local minima problem. We check the effect of bumping using Calhousing dataset. The results are in Table 4.8. The results show that bumping method efficiently reduces the local minima problem. X means without using bootstrap samples and refers to directly using the training data.

Based on test error, it appears that not using bumping leads to the occurrence of local minima problems. However, when using bumping, such issues are relatively less likely to happen, and the number of bootstrap samples(B) does not have a significant impact.

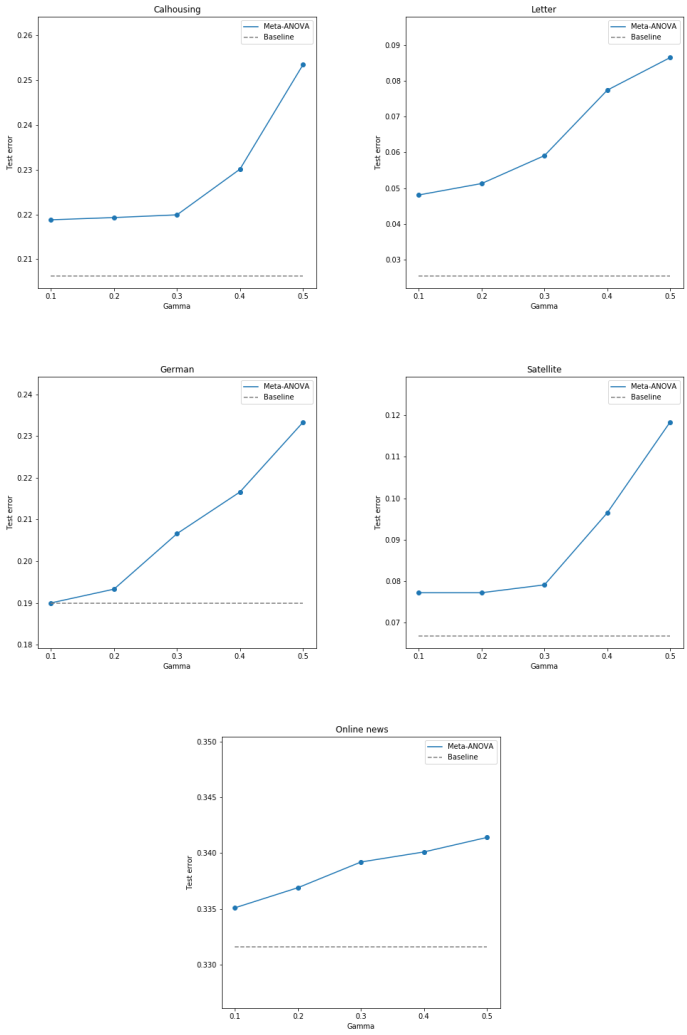

Figure 4.4: The results of sensitive analysis of γ using Real data : For each γ, we report the test errors of *Meta-ANOVA*. The dashed line represents the test error of baseline model.

Table 4.8: The results of the effect of bumping : We report the test errors for each bumping size.

Bumping size(B)	Test error
X	0.2584
5	0.2188
10	0.2179
20	0.2177

4.3.4 The effect of synthetic dataset

To better approximate the given predictive model, we use not only the training dataset but also synthetic dataset. We check the effect of synthetic dataset using Calhousing dataset. The results are in Table 4.9. The results show that synthetic dataset helps the ability to approximate the given predictive model.

Table 4.9: The results of the effect of synthetic dataset : We report the test errors for various combinations of datasets.

Dataset	Test error
Train	0.2262
Train + interpolation	0.2195
Train + interpolation + extrapolation	0.2188

Chapter 5

Conclusion

In this thesis, we introduce a new post-process interpretation method called *Meta-ANOVA*. *Meta-ANOVA* interprets a given pre-trained predictive model by approximating it as a functional ANOVA model. Since functional ANOVA model is interpretable on its own, we can interpret the given predictive model.

To interpret the given predictive model correctly, the functional ANOVA model should closely approximate the original model. To approximate the given model well, we develop the interactions detection algorithm and introduce the modified learning algorithm for *ANOVA-Boosting* to utilize higher order interactions.

In comparison with other interpretation methods, our method can detect higher order interactions the given predictive model has. Moreover, *Meta-ANOVA* can give local and global interpretation simultaneously. Also, our method is model-agnostic, which means it can be applicable as long as the outputs can be obtained.

Recently, transparent box models such as *NAM* (Neural Additive Model) [Agarwal et al., 2021] and *NBM* (Neural Basis Model) [Radenovic et al., 2022] have been developed using DNNs. It would be interesting to use them as approximation model. We leave this as future work.

Bibliography

Rishabh Agarwal, Nicholas Frosst, Xuezhou Zhang, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34, 2021.

Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

Alae Chouiekh and EL Hassane Ibn EL Haj. Convnets for fraud detection analysis. *Procedia Computer Science*, 127:133–138, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer, 1992.

Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

Ulrike Grömping. South german credit data: Correcting a widely used data set. *Rep. Math., Phys. Chem., Berlin, Germany, Tech. Rep*, 4:2019, 2019.

Chong Gu. *Smoothing spline ANOVA models*, volume 297. Springer, 2013.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jinseog Kim, Yongdai Kim, Yuwon Kim, Sunghoon Kwon, and Sangin Lee. Boosting on the functional anova decomposition. *Statistics and Its Interface*, 2(3):361–368, 2009.

Yongdai Kim, Yuwon Kim, and Jinseog Kim. Anova boosting. 2005.

Yi Lin and Hao Helen Zhang. Component selection and smoothing in multivariate nonparametric regression. 2006.

Zirui Liu, Qingquan Song, Kaixiong Zhou, Ting-Hsiang Wang, Ying Shan, and Xia Hu. Detecting interactions from neural networks via topological analysis. *Advances in Neural Information Processing Systems*, 33, 2020.

Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Filip Radenovic, Abhimanyu Dubey, and Dhruv Mahajan. Neural basis models for interpretability. *Advances in Neural Information Processing Systems*, 35:8414–8426, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " 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*, pages 1135–1144, 2016.

Lloyd S Shapley et al. A value for n-person games. 1953.

Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

Daria Sorokina, Rich Caruana, Mirek Riedewald, and Daniel Fink. Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th international conference on Machine learning*, pages 1000–1007, 2008.

Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.

Appendix A

Appendix

A.1 Proof of Theorem 3.2.1.

Note that for a given $\mathbf{j} \subset [p]$, we can represent f as following:

$$f(\mathbf{x}) = \beta_0 + \sum_{\mathbf{j}' < \mathbf{j}} G_{\mathbf{j}'\mathbf{j}}(\mathbf{x}) + G_{\mathbf{j}\mathbf{j}}(\mathbf{x}) + \sum_{\mathbf{j}_3 < \mathbf{j}^c} f_{\mathbf{j}_3}(\mathbf{x}_{\mathbf{j}_3}),$$

where $G_{\mathbf{j}'\mathbf{j}}(\mathbf{x}) = f_{\mathbf{j}'}(\mathbf{x}_{\mathbf{j}'}) + \sum_{\mathbf{j}_2 \subset \mathbf{j}^c} f_{\mathbf{j}' \cup \mathbf{j}_2}(\mathbf{x}_{\mathbf{j}' \cup \mathbf{j}_2})$.

'Only if' part :

For a given $\mathbf{j} \subset [p]$ $f_{\mathbf{j}'} = 0$ for all $\mathbf{j}' > \mathbf{j}$ means

$$f(\mathbf{x}) = \beta_0 + \sum_{\mathbf{j}' < \mathbf{j}} G_{\mathbf{j}'\mathbf{j}}(\mathbf{x}) + f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}) + \sum_{\mathbf{j}_3 < \mathbf{j}^c} f_{\mathbf{j}_3}(\mathbf{x}_{\mathbf{j}_3}),$$

and $D_{\mathbf{j}} f(\mathbf{x}) = D_{\mathbf{j}} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}})$ which does not depend on $\mathbf{x}_{\mathbf{j}^c}$. Therefore $\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{x}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) \right\} = 0$.

'If' part :

For a given $\mathbf{j} \subset [p]$, $\mathrm{Var}_{\mathbf{X}'_{\mathbf{j}^c} \sim Q_{\mathbf{j}^c}} \left\{ D_{\mathbf{j}} f(\mathbf{X}'_{\mathbf{j}}, \mathbf{X}'_{\mathbf{j}^c}) | \mathbf{X}'_{\mathbf{j}} \right\} = 0$ means that

$D_{\mathbf{j}} f(\mathbf{x})$ does not depend on $\mathbf{x}_{\mathbf{j}^c}$. Note that

$$D_{\mathbf{j}} f(\mathbf{x}) = D_{\mathbf{j}} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}) + D_{\mathbf{j}} \left[\sum_{\mathbf{j}_2 \subset \mathbf{j}^c} f_{\mathbf{j} \cup \mathbf{j}_2}(\mathbf{x}_{\mathbf{j} \cup \mathbf{j}_2}) \right] = D_{\mathbf{j}} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}}) + D_{\mathbf{j}} f_{\mathbf{j}+}(\mathbf{x}).$$

Since both $D_{\mathbf{j}} f(\mathbf{x})$ and $D_{\mathbf{j}} f_{\mathbf{j}}(\mathbf{x}_{\mathbf{j}})$ do not depend on $\mathbf{x}_{\mathbf{j}^c}$, so does $D_{\mathbf{j}} f_{\mathbf{j}+}(\mathbf{x})$. Thus, $f_{\mathbf{j}'} = 0$ for all $\mathbf{j}' > \mathbf{j}$. □

A.2 Extension to categorical input variables

Let $\mathbf{x} \in \mathcal{X} = \{0, 1\}^p$ be an input vector and $f : \mathcal{X} \to \mathbb{R}$ be a given predictive model constructed by any machine learning algorithm such as random forest, deep neural networks. We assume ANOVA model as follows

$$f(\mathbf{x}) = \beta_0 + \sum_{k=1}^{p} \left\{ \sum_{\mathbf{j} \in J_k} \beta_{\mathbf{j}} \mathbf{x}_{\mathbf{j}}! \right\} \tag{A.1}$$

Note that when x_js are all binary, i.e. 0 or 1, all $f(\mathbf{x})$ can be perfectly represented by an ANOVA model. For $\mathbf{j} \subset [p]$, $\beta_{\mathbf{j}} \neq 0$ implies that the interaction $\mathbf{j}$ is valid for the given function $f(\mathbf{x})$. For a given $\mathbf{j} \subset [p]$, we write equation (A.1) as follows:

$$f(\mathbf{x}) = \beta_0 + \sum_{\mathbf{j}' \subset \mathbf{j}} \mathbf{x}_{\mathbf{j}'}! \left\{ \beta_{\mathbf{j}'} + \sum_{\mathbf{j}_2 \subset \mathbf{j}^c} \beta_{\mathbf{j}' \cup \mathbf{j}_2} \mathbf{x}_{\mathbf{j}_2}! \right\} + \sum_{\mathbf{j}_3 \subset \mathbf{j}^c} \beta_{\mathbf{j}_3} \mathbf{x}_{\mathbf{j}_3}!.$$

For given $\mathbf{j}$ and $\mathbf{j}' \subset \mathbf{j}$, let $g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = \beta_{\mathbf{j}'} + \sum_{\mathbf{j}_2 \subset \mathbf{j}^c} \beta_{\mathbf{j}' \cup \mathbf{j}_2} \mathbf{x}_{\mathbf{j}_2}!$. Then, we can write

$$f(\mathbf{x}) = \beta_0 + \sum_{\mathbf{j}' \subset \mathbf{j}} \mathbf{x}_{\mathbf{j}'}! g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) + \sum_{\mathbf{j}_3 \subset \mathbf{j}^c} \beta_{\mathbf{j}_3} \mathbf{x}_{\mathbf{j}_3}!.$$

Theorem A.2.1. *For a given* $\mathbf{j}$, $\beta_{\mathbf{j}'} = 0$ *for all* $\mathbf{j}' > \mathbf{j}$ *if and only if* $g_{\mathbf{j},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c})$ *is a constant function for all* $\mathbf{x}_{\mathbf{j}^c}$, *where* $\mathbf{j}' > \mathbf{j}$ *means* $\mathbf{j}' \supset \mathbf{j}$ *but* $\mathbf{j}' \neq \mathbf{j}$.

Proof) The 'if' part is trivial. For the 'only if part', we can first show that $g_{\mathbf{j}'\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = \beta_{\mathbf{j}}$ when $\mathbf{x}_{\mathbf{j}^c} = \mathbf{0}$. In turn, we can show that $\beta_{\mathbf{j}'} = 0$ for any $\mathbf{j}' > \mathbf{j}$ and $|\mathbf{j}' - \mathbf{j}| = 1$ by let $\mathbf{x}_{\mathbf{j}^c}$ such that $x_j = 1$ for $j \in \mathbf{j}' - \mathbf{j}$ and 0 otherwise. By applying similar arguments repeatedly, we can show that $\beta_{\mathbf{j}'} = 0$ for all $\mathbf{j}' > \mathbf{j}$. □

Theorem A.2.2. *For any* $\mathbf{j}$, *we can represent* $g_{\mathbf{j},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c})$ *as follow.*

$$g_{\mathbf{j},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = \sum_{\mathbf{j}' \subseteq \mathbf{j}} (-1)^{|\mathbf{j}-\mathbf{j}'|} f(\mathbf{x} : \mathbf{x}_{\mathbf{j}'} = 1, \mathbf{x}_{\mathbf{j}-\mathbf{j}'} = 0) \tag{A.2}$$

Proof) Let $\mathbf{j} = (j_1, ..., j_k)$ where $k = |\mathbf{j}|$. Since

$$f(\mathbf{x} : \mathbf{x}_j = 1) = \beta_0 + g_{\mathbf{j},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) + \sum_{\mathbf{j}' < \mathbf{j}} g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) + \sum_{\mathbf{j}_3 \subset \mathbf{j}^c} \beta_{\mathbf{j}_3} \mathbf{x}_{\mathbf{j}_3}! \text{ and}$$
$$f(\mathbf{x} : \mathbf{x}_j = 0) = \beta_0 + \sum_{\mathbf{j}_3 \subset \mathbf{j}^c} \beta_{\mathbf{j}_3} \mathbf{x}_{\mathbf{j}_3}!,$$

the following holds:

$$g_{\mathbf{j},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = f(\mathbf{x} : \mathbf{x}_j = 1) - f(\mathbf{x} : \mathbf{x}_j = 0) - \sum_{\mathbf{j}' < \mathbf{j}} g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}).$$

Considering that

$$f(\mathbf{x} : \mathbf{x}_{\mathbf{j}'} = 1, \mathbf{x}_{\mathbf{j}-\mathbf{j}'} = 0) = \beta_0 + \sum_{\tilde{\mathbf{j}} \subseteq \mathbf{j}'} g_{\tilde{\mathbf{j}},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) + \sum_{\mathbf{j}_3 \subset \mathbf{j}^c} \beta_{\mathbf{j}_3} \mathbf{x}_{\mathbf{j}_3}!, \tag{A.3}$$

we can get

$$\sum_{\tilde{\mathbf{j}} \subseteq \mathbf{j}'} g_{\tilde{\mathbf{j}},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = f(\mathbf{x} : \mathbf{x}_{\mathbf{j}'} = 1, \mathbf{x}_{\mathbf{j}-\mathbf{j}'} = 0) - f(\mathbf{x} : \mathbf{x}_j = 0). \tag{A.4}$$

By the principle of inclusion-exclusion, we can represent $\sum_{\mathbf{j}'<\mathbf{j}} g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c})$ given as:

$$\begin{aligned}
\sum_{\mathbf{j}'<\mathbf{j}} g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = (-1)^0 & \sum_{1\le i_1\le k} \sum_{\tilde{\mathbf{j}}\subset \mathbf{j}-\{j_{i_1}\}} g_{\tilde{\mathbf{j}},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) \\
+ (-1)^1 & \sum_{1\le i_1<i_2\le k} \sum_{\tilde{\mathbf{j}}\subset \mathbf{j}-\{j_{i_1},j_{i_2}\}} g_{\tilde{\mathbf{j}},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) \\
+ & \cdots \\
+ (-1)^{k-2} & \sum_{1\le i_1<i_2<\cdots<i_{k-1}\le k} \sum_{\tilde{\mathbf{j}}\subset \mathbf{j}-\{j_{i_1},\ldots,j_{i_{k-1}}\}} g_{\tilde{\mathbf{j}},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c})
\end{aligned} \tag{A.5}$$

By (A.4) and (A.5) we have the following:

$$\begin{aligned}
\sum_{\mathbf{j}'<\mathbf{j}} g_{\mathbf{j}',\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c}) = (-1)^0 & \sum_{1\le i_1\le k} f(\mathbf{x}:\mathbf{x}_{\mathbf{j}-\{j_{i_1}\}}=1,\mathbf{x}_{\{j_{i_1}\}}=0) \\
+ (-1)^1 & \sum_{1\le i_1<i_2\le k} f(\mathbf{x}:\mathbf{x}_{\mathbf{j}-\{j_{i_1},j_{i_2}\}}=1,\mathbf{x}_{\{j_{i_1},j_{i_2}\}}=0) \\
+ & \cdots \\
+ (-1)^{k-2} & \sum_{1\le i_1<i_2<\cdots<i_{k-1}\le k} f(\mathbf{x}:\mathbf{x}_{\mathbf{j}-\{j_{i_1},\ldots,j_{i_{k-1}}\}}=1,\mathbf{x}_{\{j_{i_1},\ldots,j_{i_{k-1}}\}}=0) \\
- & \begin{cases} 0 & \text{if } k \text{ is odd} \\ 2f(\mathbf{x}:\mathbf{x}_{\mathbf{j}}=0) & \text{if } k \text{ is even} \end{cases}
\end{aligned} \tag{A.6}$$

Thus by combining (A.3) and (A.6), the proof is done. □

Note that $g_{\mathbf{j},\mathbf{j}}(\mathbf{x}_{\mathbf{j}^c})$ can be defined by partial ordinal derivative when the input variables can take value $\{0,1\}$. Thus, partial derivative can be used when categorical input variables are given.

A.3 Additional results for Simulated data

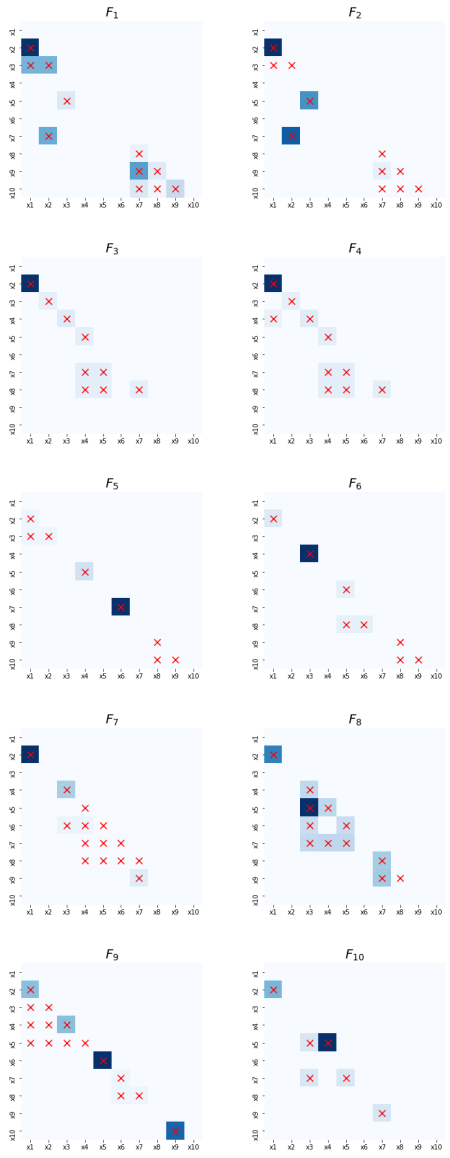

Figure A.1: Heat maps of pair interaction scores proposed by Meta ANOVA framework for synthetic function $F_1 - F_{10}$. Cross-marks indicate ground truth interactions.

A.4 Additional results for Real data

Table A.1: Calhousing : The number of detected interactions for various γ.

	$\gamma : 0.1$	$\gamma : 0.2$	$\gamma : 0.3$	$\gamma : 0.4$	$\gamma : 0.5$
Main	8	8	8	8	8
Second	21	21	21	21	15
Third	35	30	16	6	0
Fourth	20	5	2	0	0

Table A.2: Letter : The number of detected interactions for various γ.

	$\gamma : 0.1$	$\gamma : 0.2$	$\gamma : 0.3$	$\gamma : 0.4$	$\gamma : 0.5$
Main	16	16	16	16	16
Second	120	105	78	55	36
Third	50	50	27	10	0
Fourth	0	0	0	0	0

Table A.3: German : The number of detected interactions for various γ.

	$\gamma : 0.1$	$\gamma : 0.2$	$\gamma : 0.3$	$\gamma : 0.4$	$\gamma : 0.5$
Main	61	61	61	61	61
Second	300	253	91	21	10
Third	7	6	2	1	0
Fourth	0	0	0	0	0

Table A.4: Satellite : The number of interactions for various γ.

	$\gamma : 0.1$	$\gamma : 0.2$	$\gamma : 0.3$	$\gamma : 0.4$	$\gamma : 0.5$
Main	36	36	36	36	36
Second	300	300	210	91	45
Third	100	99	7	1	1
Fourth	3	3	1	0	0

Table A.5: Online news : The number of interactions for various γ.

	$\gamma : 0.1$	$\gamma : 0.2$	$\gamma : 0.3$	$\gamma : 0.4$	$\gamma : 0.5$
Main	58	58	58	58	58
Second	276	91	36	15	10
Third	1	1	1	1	1
Fourth	0	0	0	0	0

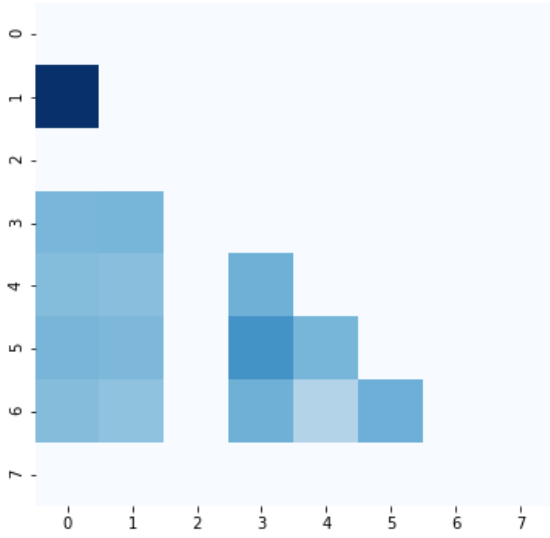

Figure A.2: Calhousing : Heat map of 2nd-order interaction scores proposed by Meta-ANOVA framework.

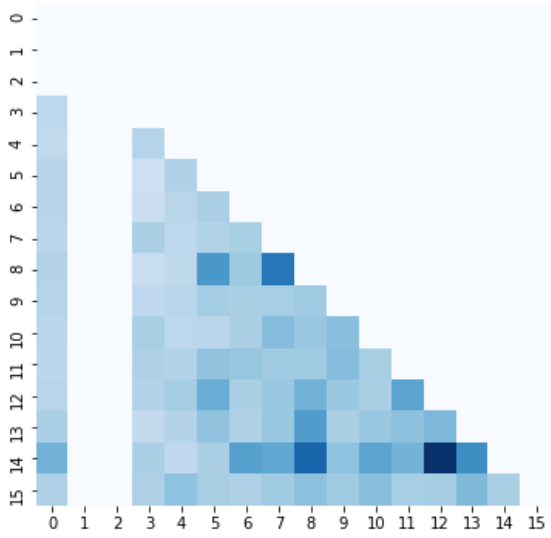

Figure A.3: Letter : Heat map of 2nd-order interaction scores proposed by Meta-ANOVA framework.

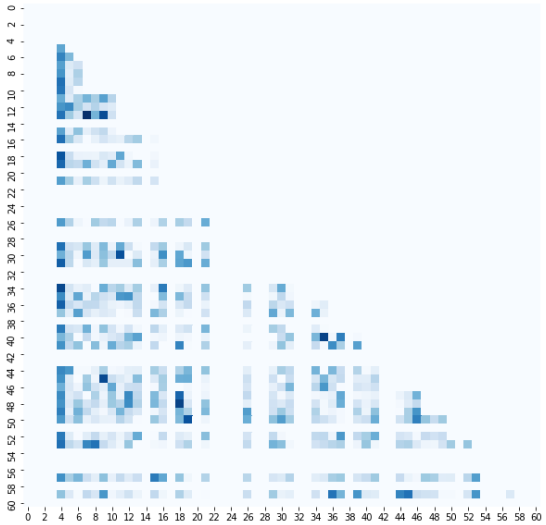

Figure A.4: German : Heat map of 2nd-order interaction scores proposed by Meta-ANOVA framework.

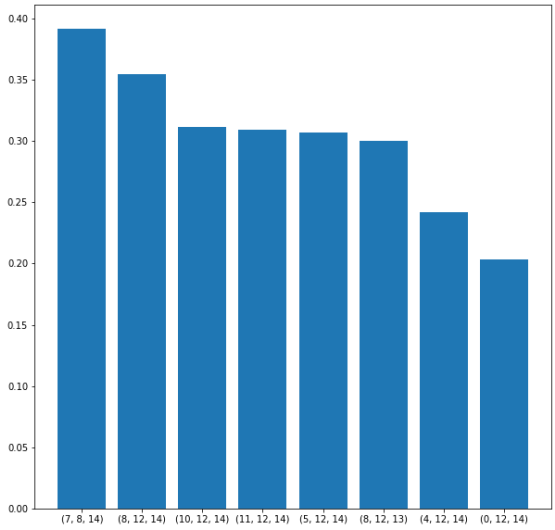

Figure A.5: Letter : Barplot of 3rd-order interaction strength scores proposed by Meta-ANOVA framework.

A.5 Comparison with local interpretation models

We compare our method to *LIME* and *SHAP-values*, which are local interpretation models. They utilize only main feature to approximate models. When *LIME* and *SHAP-values* are applied to a model in which the roles of interactions are important (e.g., tree-based models), there could be bias in the estimated coefficients

since they only utilize main features to approximate the models. To investigate this issue, we conduct experiments using two synthetic models : 1) $F(\mathbf{x}) = 2x_1 + 3x_2 + 10x_1x_2$ and 2) F_3 in Table 4.1. Inputs for F are sampled from $Unif(-1, 1)$

For *LIME* and *SHAP-values*, we randomly sample 20 inputs and apply each input to *LIME* and *SHAP-values*. Next, we obtain 20 coefficients forx_2 and x_8, respectively, where each of which is linear in synthetic function F and F_3. We use averages of 20 coefficients for *LIME* and *SHAP-values*. The results are summarized in Figure A.6. We can clearly see that *Meta-ANOVA* provides well-estimated coefficients for each two variables. In contrast, *LIME* and *SHAP-values* become worse for this setting and even show negative coefficient for some inputs(true is positive for both models). This shows that local explanation should be interpreted carefully.

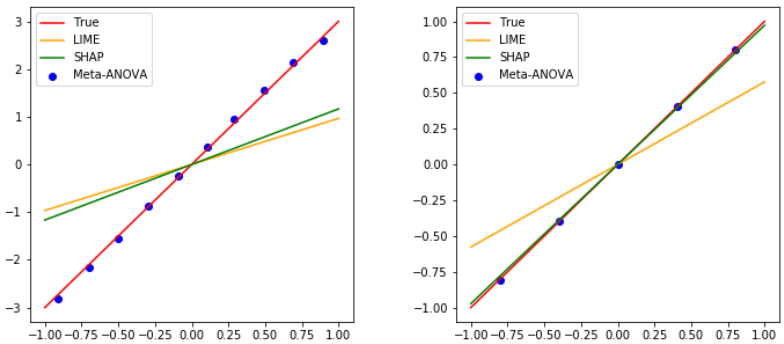

Figure A.6: The comparison of estimated coefficient for synthetic function F(**Left**) and F_3(**Right**) between *Meta-ANOVA* and local surrogate models. We check the coefficient of X_2 and X_9 for F and F_3 respectively.

국문초록

본 학위 논문에서는 새로운 post-process interpretation 방법인 *Meta-ANOVA*를 제안한다. 최근 다양한 분야에서 머신러닝 모형이 주목할만큼 좋은 예측 성능을 보이고 있다. 그러나 머신러닝 모형의 좋은 예측 성능에도 불구하고 최근에 개발된 머신러닝 모형은 일반적으로 설명력이 부족하다. 그 이유는 좋은 예측 성능을 보이는 모형들은 모형의 복잡도 또한 높기 때문이다. 이러한 복잡도는 사람들이 해당 모형을 이해하기 어렵고 따라서 다양한 분야에서 사용되는 것을 방해한다.

따라서 우리는 복잡한 모형을 해석하기 위해 *Meta-ANOVA*라는 새로운 post-process 방법론을 제안하였다. 주어진 모형을 이해하고 설명하기 위해 우리는 해당 모형이 가지고 있는 교호작용을 찾는 새로운 알고리즘을 제안하였다. 뿐만 아니라 우리는 모형이 가진 교호작용을 찾은 뒤 이를 이용하여 주어진 모형을 효과적으로 학습하는 새로운 functional ANOVA model의 학습 알고리즘을 제안하였다.

주요어: 설명가능 AI, 모형 설명, 교호작용 탐지, Functional ANOVA model, Identifiability

학 번: 2016–20277