Shapley Values of Structured Additive Regression Models and Application to RKHS Weightings of Functions

Anonymous authors
Paper under double-blind review

Abstract

The ability to interpret machine learning models is proving more and more invaluable, as their use in sensitive domains requires trust. Therefore, work to improve explanation methods, especially the interpretation of complex models, is of high importance. With this in mind, the purpose of this paper is twofold. First, we present an algorithm for efficiently calculating the Shapley values of a family of models, Structured Additive Regression (STAR) models, which allow more variable interactions than Generalized Additive Models (GAMs). Second, we present a new instantiation in the RKHS Weightings of Functions paradigm, better adapted to regression, and show how to transform it and other RKHS Weightings instantiations into STAR models. We therefore introduce a new family of STAR models, as well as the means to interpret their outputs in a timely manner.

1 Introduction

The ability to interpret a machine learning model is an increasingly important consideration. Highly complex models can achieve great accuracy, but understanding their inner workings is challenging. Why did the model offer this particular prediction for this input? What features of the data was the decision based on? On the other hand, simpler models, like Generalized Additive Models (GAMs, Hastie (2017)) or decision trees, are inherently easier to understand, but do not always reach the same levels of performance (though recent work appears to show otherwise (Rudin et al., 2024)). Work to improve explanation methods, especially the interpretation of more complex models, can help bridge this gap.

Informally, we mean by *interpreting a model* the task of explaining its output. In sensitive contexts such as healthcare (Caruana et al., 2015), energy systems (Zhang & Chen, 2024) or machine fault diagnosis (Brusa et al., 2023), to give only a few examples, mistakes can be very costly, and so the ability to understand, and thus trust, an AI model is crucial. In this paper, we specifically consider the **attribution problem**, which consists in distributing the prediction score between the data features (Sundararajan & Najmi, 2020). On one end of the spectrum of interpretability lie linear models, where the contribution of each variable is obvious and independent of all other variables. On the other end, we have large deep neural networks, the incredibly complex inner workings of which are singularly difficult to elucidate (Montavon et al., 2018).

Post-hoc interpretability methods attempt to explain even complex models through various means. Just some examples are LIME (Ribeiro et al., 2016), which explains the prediction of a complex model by approximating the model locally around the instance using a simple predictor, and Partial Dependance Plots (Friedman, 2001), which visualize the effect of a variable by holding others constant.

Another widely used tool for interpreting models, and the subject of this paper, is the Shapley value (see e.g. Winter (2002)), originally devised in the context of game theory (Shapley et al., 1953). Adapted to the machine or statistical learning context, it can be used to assess the contribution of each variable to the output of a model, for a given example. Additionally, it can be averaged over a dataset to see the relative importance of each variable.

With all of this in mind, the purpose of this paper is twofold. First, we present an algorithm for efficiently calculating the Shapley values of a large family of prediction models, Structured Additive Regression (STAR) models. STAR models are a further generalization of Generalized Additive Models (GAMs) (Hastie, 2017) that allow variable interactions, making them potentially much more expressive models. This algorithm will thus allow the interpretation of much more complex models, making them more viable in sensitive domains.

Second, we present a new instantiation in the RKHS Weightings of Functions paradigm (Dubé & Marchand, 2024), one better suited to regression. Lastly, we show how to transform it and other RKHS Weightings into STAR models. By introducing this new family of STAR models, we seek to increase the use of this now interpretable family of models.

The rest of the paper is as follows. Section 2 covers Shapley values and RKHS Weightings of Functions. Section 3 presents the formula and corresponding algorithm for calculating the Shapley values of a STAR model. Section 4 presents a new RKHS Weighting instantiation, and shows how to transform an RKHS Weighting model into a STAR model. Section 5 contains three experiments, showing the regression and time series performance of the models introduced in this paper, and the computation time of the Shapley values using our new algorithm. Finally, we go over some limitations of this paper, as well as future work to be done, in Section 6, and conclude in Section 7.

2 Review of Shapley values and RKHS Weightings

The two main subjects of this paper are Shapley values and RKHS Weightings, two tools we use within the context of machine learning. We start in the next section by defining the basic machine learning notions that we use in this paper. Then, we present our definition of the Shapley value, and we end on a condensed summary of RKHS Weightings.

2.1 Machine learning

The **regression** setting in machine learning consists of learning to predict real values from examples. Given an **instance space** $\mathcal{X}$, usually $\mathcal{X} = \mathbb{R}^n$, the **label space** $\mathcal{Y} = \mathbb{R}$, and a **dataset** of examples $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$, our task is to find a **predictor** (also called **model**) $h : \mathcal{X} \to \mathcal{Y}$ that optimizes some objective function of the sample data. To measure the quality of a model, we use a **loss function** $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which in this paper will be the **squared loss**:

$$\ell(h(x), y) := \frac{1}{2}(h(x) - y)^2. \tag{1}$$

The **empirical risk** is the sample average loss. When using the squared loss, this is called the **Mean Square Error** (MSE):

$$\mathrm{MSE}_{\mathcal{S}}(h) := \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2. \tag{2}$$

The MSE has units of the labels squared. To successfully compare experimental results on different datasets, we will instead report the **coefficient of determination**:

$$R^2_{\mathcal{S}}(h) := 1 - \frac{\sum_{i=1}^m (y_i - h(x_i))^2}{\sum_{i=1}^m (y_i - \bar{y})^2}, \tag{3}$$

where $\bar{y} := \frac{1}{m} \sum_{i=1}^m y_i$ is the average label.

In practice, simply minimizing the sample MSE is insufficient. We seek a model that will successfully generalize to new, yet-to-be-seen instances. Indeed, we can assume that the data comes from some unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The true risk, or true MSE, is the expected squared loss over this distribution:

$$\mathrm{MSE}_{\mathcal{D}}(h) := \frac{1}{2} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[(h(x) - y)^2\right]. \tag{4}$$

We estimate this value by calculating the MSE on a set of examples distinct from the sample.

A **learning algorithm** $\mathcal{A}$ is a function that takes in a sample $\mathcal{S}$, and returns a predictor, i.e. $\mathcal{A}(\mathcal{S}) = h(x)$. It is this algorithm that seeks to minimize the true risk by optimizing a function of the best available proxy, the empirical MSE. Most learning algorithms have **hyperparameters**; the number of trees in a random forest; the learning rate of a neural network; the regularization parameter of the kernel ridge regression. Each combination of parameters yields an algorithm. Suppose we have C combinations to choose from. This implies we have the candidate algorithms $\mathcal{A}_1$ to $\mathcal{A}_C$. To select the best one, we will use **cross-validation**, more specifically k-fold cross-validation. The sample $\mathcal{S}$ is first divided into k equal disjoint subsets $\mathcal{S}_1$ to $\mathcal{S}_k$. Each algorithm $\mathcal{A}_c$ is applied k times, giving us the k models $\mathcal{A}_c(\mathcal{S} \setminus \mathcal{S}_1), \ldots, \mathcal{A}_c(\mathcal{S} \setminus \mathcal{S}_k)$. We can estimate the true risk of each of these models through its MSE on the holdout set, e.g. $\mathrm{MSE}_{\mathcal{D}}(\mathcal{A}_c(\mathcal{S} \setminus \mathcal{S}_1)) \approx \mathrm{MSE}_{\mathcal{S}_1}(\mathcal{A}_c(\mathcal{S} \setminus \mathcal{S}_1))$. In turn, we can estimate the true risk of $\mathcal{A}_c(\mathcal{S})$ (the model trained on the entire dataset) as the average MSE of these k models:

$$\mathrm{MSE}_{\mathcal{D}}(\mathcal{A}_c(\mathcal{S})) \approx \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_{\mathcal{S}_i}(\mathcal{A}_c(\mathcal{S} \setminus \mathcal{S}_i)). \tag{5}$$

The algorithm with the lowest approximated MSE is chosen, and trained again on the entire sample.

2.2 Shapley values

To define the Shapley value, we require some new notation. Consider the instance space $\mathcal{X} = \mathbb{R}^n$, and denote $[n] := \{1, \ldots, n\}$ the natural numbers up to n, identifying the n variables that make up an example $x = (x_1, \ldots, x_n) \in \mathcal{X}$. A bijective function $\pi : [n] \to [n]$ is called a **permutation**; $\pi(i)$ is the position of variable i in the permutation. Also define:

$$\pi_{:i} := \left\{ j \in [n] \;\middle|\; \pi(j) < \pi(i) \right\} \tag{6}$$

the set of variables in π up to (but excluding) i. Let Ω be the set of all permutations of $[n]$. Finally, we need to define the **replacement function**. For some feature subset $P \subseteq [n]$, and $x, z \in \mathcal{X}$, the vector $r_P(z, x)$ is z, except that the variables in P have been replaced by the values in x:

$$r_P(z, x)_i = \begin{cases} z_i & i \notin P \\ x_i & i \in P \end{cases} \tag{7}$$

Given these notations and the background dataset $\mathcal{S} \subset \mathcal{X}$, the **Shapley value** of variable i, for example x, for the predictor h, is defined as:

$$\phi_i^{\mathrm{SHAP}}(h, x) := \mathop{\mathbb{E}}_{z \sim \mathcal{S}} \mathop{\mathbb{E}}_{\pi \sim U(\Omega)} \left[h(r_{\pi_{:i} \cup \{i\}}(z, x)) - h(r_{\pi_{:i}}(z, x)) \right]. \tag{8}$$

(Definition taken from Laberge et al. (2023).)

It is the excess contribution of variable i to the model prediction for the example x, compared to its average contribution on the dataset (or background) $\mathcal{S}$. The Shapley value is the unique attribution method that satisfies four key properties. We list them informally here (inspired by Fryer et al. (2021)):

1. **Efficiency.** The Shapley values of an example x sum to the **gap**, i.e. the excess output $h(x) - \mathbb{E}_{z \sim \mathcal{S}} h(z)$ over the average. From the point of view of game theory, we say that the gain is distributed among all the agents (the variables).

2. **Symmetry.** Two variables with equal contributions have equal Shapley values.

3. **Null player.** A variable that has no contribution has a Shapley value of 0.

4. **Linearity.** The Shapley values of a sum of functions $h(x) + g(x)$ are the sum of the Shapley values of the individual functions.

These properties make the Shapley value a uniquely interesting attribution method for the contribution of each variable to the prediction of a model, although some raise doubts on its reliability (Huang & Marques-Silva, 2023) or the desirability of these four properties (Fryer et al., 2021). We do not weigh in on this debate in this paper.

Calculating Equation (8) directly is in $\mathcal{O}(n!)$, as it requires enumerating every permutation of $[n]$. It therefore can't reasonably be used except on particularly low-dimensional datasets. However, some model families admit much faster formulas or algorithms for calculating the Shapley values. Such model families include linear models (Lundberg & Lee, 2017), Generalized Additive Models (GAMs), and Trees and ensembles of Trees (Lundberg et al., 2018). The common characteristic of these models is limited variable interactions, so that the contribution of a particular variable can be calculated independently of the variables with which it does not interact. For example, the Shapley values of a linear model $h(x) = \sum_j w_j x_j$ are obtained easily from its coefficients by the formula $\phi_i^{\text{SHAP}}(h, x) = w_i(x_i - \mathbb{E}_{z\sim\mathcal{S}}\, z_i)$ (Lundberg & Lee, 2017); the Shapley values of a GAM $h(x) = \sum_j f_j(x_j)$ are given by:

$$\phi_i^{\text{SHAP}}(h, x) = f_i(x_i) - \mathop{\mathbb{E}}_{z\sim\mathcal{S}}[f_i(z_i)]. \tag{9}$$

(Proof in the appendix.)

Structured Additive Regression (STAR) models are a generalization of both linear models and GAMs (see e.g. Mayer et al. (2021)). A STAR model allows variable interactions by defining the predictor over subsets of the data features:

$$h(x) = \sum_{I\subseteq[n]} f_I(x_I), \tag{10}$$

where x_I is the partial vector $(x_{I_1}, \ldots, x_{I_{|I|}})$, and f_I is a function defined over the feature subset I. Importantly, not all of these functions f_I need to be nonzero. We can indeed choose which variable interactions are desired by selecting which feature subsets I will be associated with a nonzero f_I. For example, GAMs use only singleton feature subsets, i.e. individual variables, while GA2Ms (Lou et al., 2013) use up to pairwise interactions between the variables.

STAR models have various names in the literature. Bordt & von Luxburg (2023) define Generalized Additive Models of order k as models of the form:

$$h(x) = \sum_{I\subseteq[n], |I|\leq k} f_I(x_I), \tag{11}$$

and introduce k-Shapley Values, which attribute a contribution to each feature subset, instead of only individual variables. This generalizes Shapley values, but suffers from the same high computational cost. Laberge et al. (2024) call Equation (10) a Functional Decomposition, following previous work on the subject (Hooker, 2004; Kuo et al., 2010), and show among other things how to obtain the Shapley values of a function from its Interventional Decomposition (Kuo et al., 2010). This, again, is a high cost procedure.

In Section 3, we present an algorithm for calculating the Shapley values of any STAR model. As long as the maximal number of interactions k is limited, this is an efficient algorithm. Specifically, its computational complexity scales exponentially in the maximal size k of the feature subsets, but is only linear in the dimensionality n of the data. This allows using many more interaction terms, while maintaining the ability to interpret the model predictions through its Shapley values.

2.3 RKHS weightings of functions

Dubé & Marchand (2024) introduced a new, flexible model family, parameterized through the tuple $(\mathcal{W}, \phi, \mathcal{K}, p)$ of a parameter space $\mathcal{W}$; base predictor $\phi : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$; kernel $\mathcal{K} : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$; distribution p over $\mathcal{W}$. They defined the operator $\Lambda : \mathcal{H} \to L^\infty(\mathcal{X})$, which takes as input a weight function $\alpha \in \mathcal{H}$, and returns the bounded predictor $\Lambda\alpha : \mathcal{X} \to \mathbb{R}$ via the equation:

$$\Lambda\alpha(x) := \mathop{\mathbb{E}}_{w\sim p}[\alpha(w)\phi(w, x)]. \tag{12}$$

This is a highly flexible family of models, and we will show in Section 4 how to build RKHS Weightings that are also STAR models. These models will be interpretable, as their Shapley values can be calculated using Algorithm 1, which we introduce in the next section.

Dubé & Marchand (2024) presented two instantiations of the model, which they called I1 and I2. We will refer to these as RWSign and RWStumps respectively (RW standing for RKHS Weighting). While these instantiations can be used for regression, they are first of all intended for classification, since the base predictor, in both cases, can only take two values. We introduce in Section 4 a new instantiation better suited to regression, using the ReLU as the base predictor. All three instantiations can then be converted to STAR models using the procedure we show in Section 4.

3 Shapley values of STAR models

In this section, we present the formula and corresponding algorithm for calculating the Shapley values of Structured Additive Regression (STAR) models. We begin with the formula, encapsulated in the following theorem.

Theorem 3.1 (Shapley values of a STAR model)**.** *Consider a STAR model:*

$$h(x) = \sum_{I \subseteq [n]} f_I(x_I). \tag{13}$$

Consider the replacement function r_P defined in Equation (7)*, and the sets:*

$$\mathcal{A}^+(i, I) := \{A \subset I | i \in A, 1 \leq |A| < |I|\} \tag{14}$$

$$\mathcal{A}^-(i, I) := \{A \subseteq I | i \in A, 1 < |A| \leq |I|\}. \tag{15}$$

Then the Shapley values of the model are:

$$\phi_i^{\text{SHAP}}(h, x) = \mathop{\mathbb{E}}_{z \sim \mathcal{S}} \left[\sum_{I : f_I \neq 0, i \in I} \left(\frac{f_I(x_I) - f_I(z_I)}{|I|} + \sum_{A \in \mathcal{A}^+(i,I)} \frac{f_I(r_A(z,x)_I)}{|A|\binom{|I|}{|A|}} - \sum_{A \in \mathcal{A}^-(i,I)} \frac{f_I(r_{A \setminus \{i\}}(z,x)_I)}{|A|\binom{|I|}{|A|}} \right) \right]. \tag{16}$$

Proof. The proof is a series of combinatorial arguments, and can be found in the appendix. □

Assuming that the feature subsets I are limited in size, then Equation (16) is vastly more efficient to calculate than the definition of the Shapley values in Equation (8). Instead of going through all the permutations of $[n]$, Equation (16) requires only enumerating the subsets of each feature subset. Letting $m = |\mathcal{S}|$, N the number of feature subsets I such that $f_I \neq 0$, and k the maximal size of these feature subsets, then this formula can be calculated in $\mathcal{O}(mN2^k)$, independently of the dimensionality of the data.

We should note that using Equation (16) as it is to calculate all the Shapley values for a given instance will lead to many redundant computations. Indeed, observe that:

1. For any $j \in A \in \mathcal{A}^+(i, I)$, we also have $A \in \mathcal{A}^+(j, I)$. This means that the computation $f_I(r_A(z, x)_I)$ appears in the calculation of the Shapley value of each variable in A.

2. Consider any $A \in \mathcal{A}^+(i, I)$, and a variable $j \in I \setminus A$. Then $A \cup \{j\} \in \mathcal{A}^-(j, I)$, and so the computation $f_I(r_{(A \cup \{j\}) \setminus \{j\}}(z, x)_I)$ is equal to $f_I(r_A(z, x)_I)$.

In short, when calculating the Shapley values of all variables (rather than a single one), the same computation $f_I(r_A(z, x)_I)$ is done exactly $|I|$ times; $|A|$ times from the first observation, and $|I| - |A|$ times from the second observation. We can save a factor $|I|$ by doing the computation only once. The term $\frac{f_I(x_I) - f_I(z_I)}{|I|}$ can also be calculated once, as the value is the same for all variables. These caveats in mind, we present Algorithm 1 for calculating the Shapley values of a STAR model.

Algorithm 1 STAR-SHAP: Shapley Values of a Structured Additive Regression Model

```
input Input space X ∈ ℝ^n, STAR model h(x) = Σ_{I⊆[n]} f_I(x_I), sample S
  Initialize Φ = (φ_1, ..., φ_n) = 0 ∈ ℝ^n
  for z ∈ S, I ⊆ [n] such that f_I ≠ 0 do
    baseline ← (f_I(x_I) − f_I(z_I)) / |I|
    for i ∈ I do
      φ_i ← φ_i + baseline
    end for
    for A ⊆ I, A ≠ ∅ do
      term ← f_I(r_A(z, x)_I)
      for i ∈ I do
        if i ∈ A and A ∈ A⁺(i, I) then
          φ_i ← φ_i + term / (|A| C(|I|, |A|))
        else if i ∈ I \ A and (A ∪ {i}) ∈ A⁻(i, I) then
          φ_i ← φ_i − term / (|A+1| C(|I|, |A+1|))
        end if
      end for
    end for
  end for
output (1/|S|) Φ
```

	RWSign	RWRelu	RWStumps
$\mathcal{W}$	$\mathbb{R}^n$	$\mathbb{R}^n$	$\{1,\dots,n\} \times \mathbb{R}$
$\phi(w,x)$	$\operatorname{sign}(\langle w,x\rangle)$	$\max(0,\langle w,x\rangle)$	$\operatorname{sign}(x_{w_1} - w_2)$
$\mathcal{K}(u,w)$	$\exp\left(-\frac{\|u-w\|_2^2}{2\gamma^2}\right)$	$\exp\left(-\frac{\|u-w\|_2^2}{2\gamma^2}\right)$	$\mathbb{1}[u_1 = w_1] \exp\left(-\frac{(u_2-w_2)^2}{2\gamma^2}\right)$
p	$\mathcal{N}(0,\sigma^2 I_n)$	$\mathcal{N}(0,\sigma^2 I_n)$	$\mathcal{U}(\{1,\dots,n\}) \times \mathcal{N}(0,\sigma^2)$

Table 1: Three RKHS Weightings instantiations.

4 Structured Additive RKHS Weightings of Functions

In this section, we begin by introducing a new RKHS Weighting instantiation better suited to regression than those found in Dubé & Marchand (2024). Then, we show how to transform any generic instantiation of a certain form into a Structured Additive Regression (STAR) model.

4.1 ReLU instantiation

An RKHS Weighting instantiation is a tuple $(\mathcal{W}, \phi, \mathcal{K}, p)$ of a parameter space $\mathcal{W}$, base predictor ϕ, kernel $\mathcal{K}$ (implying an RKHS $\mathcal{H}$) and distribution p. For example, instantiation I1 of Dubé & Marchand (2024) (which we call RWSign) uses $\mathcal{W} = \mathcal{X} = \mathbb{R}^n$, $\phi(w,x) = \operatorname{sign}(\langle w,x\rangle)$, the gaussian kernel $\mathcal{K}(u,w) = \exp\left(-\frac{\|u-w\|_2^2}{2\gamma^2}\right)$, and a normal distribution $p = \mathcal{N}(0,\sigma^2 I_n)$.

The instantiation that we propose, which we name RWRelu, consists simply in replacing the base predictor of RWSign by the ReLU, i.e. $\phi(w,x) = \max(0,\langle w,x\rangle)$. (See Table 1 for a quick overview of each instantiation.) In order to calculate the output of the model, which will take the form:

$$\Lambda\alpha(x) = \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{w\sim p}[\mathcal{K}(w_t,w)\phi(w,x)], \tag{17}$$

for some $\alpha = \sum_{t=1}^{T} a_t \mathcal{K}(w_t, \cdot) \in \mathcal{H}$, we need an analytical form for the expectation in Equation (17). We encapsulate that formula in the following theorem.

Theorem 4.1. *Consider the RKHS Weighting instantiation* $(\mathcal{W}, \phi, \mathcal{K}, p)$ *where* $\mathcal{W} = \mathcal{X} = \mathbb{R}^n$, $\phi(w, x) = \max(0, \langle w, x\rangle)$, $\mathcal{K}(u, w) = \exp\left(-\frac{\|u-w\|_2^2}{2\gamma^2}\right)$, $p = \mathcal{N}(0, \sigma^2 I_n)$*. Then we have:*

$$\mathop{\mathbb{E}}_{w\sim p}\left[\mathcal{K}(u, w)\phi(w, x)\right] = \left(1 + \frac{\sigma^2}{\gamma^2}\right)^{-n/2} e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}} \frac{\|x\|}{2\sqrt{\pi}} \left(\sqrt{2}\zeta e^{-\frac{\langle u', x\rangle^2}{2\zeta^2\|x\|^2}} + \sqrt{\pi}\frac{\langle u', x\rangle}{\|x\|}\left[1 + \operatorname{erf}\left(\frac{\langle u', x\rangle}{\sqrt{2}\zeta\|x\|}\right)\right]\right), \quad (18)$$

where ζ *is defined by the relationship* $\frac{1}{2\zeta^2} = \frac{1}{2\gamma^2} + \frac{1}{2\sigma^2}$ *and* $u' := \left(1 + \frac{\gamma^2}{\sigma^2}\right)^{-1} u$*.*

Proof. The calculus can be found in the appendix. □

Dubé & Marchand (2024) also define some instantiation-dependent theoretical constants which are then used in the theoretical guarantees. We provide those values for RWRelu and the relevant calculus in the appendix.

4.2 RKHS Weightings as STAR models

Instantiating an RKHS Weighting model consists in choosing its four components $(\mathcal{W}, \phi, \mathcal{K}, p)$. We will show how to instantiate an RKHS Weighting which is also a STAR model, first in full generality, and then by transforming an already existing instantiation into a STAR model. By having a STAR model which is also an RKHS Weighting, we keep all of the benefits of the RKHS Weightings, i.e. its theoretical guarantees, learning algorithms, pruning methods (Dubé & Marchand, 2024), but also gain interpretability through the computation of the Shapley values, using Algorithm 1.

The defining characteristic of STAR models is that they are built with functions that use only a subset of the variables at a time. The key to defining an RKHS Weighting which is also a STAR model is therefore the base predictor ϕ. By using a base predictor which can flexibly change the features that it considers, we will obtain a STAR model. Let's go through each ingredient and see how they must be defined.

1. A parameter space $\mathcal{W}$ will be accompanied by the space of feature subsets $\mathcal{P}([n])$ (the family of all subsets of $[n]$).
2. The base predictor will take as input both a parameter w and a feature subset I, as well as an instance x; $\phi((w, I), x) := \phi_I(w, x_I)$ could be any predictor defined on the feature subset I.
3. The kernel must take into account the feature subsets, meaning it is now a function of the form $\mathcal{K} : (\mathcal{W} \times \mathcal{P}([n])) \times (\mathcal{W} \times \mathcal{P}([n])) \to \mathbb{R}$.
4. The distribution p on $\mathcal{W} \times \mathcal{P}([n])$ will generate both parameters and feature subsets.

All in all, the instantiation will take the form $(\mathcal{W} \times \mathcal{P}([n]), \phi, \mathcal{K}, p)$. Now, to show that this will indeed be a STAR model. Given a weight function $\alpha = \sum_{t=1}^{T} a_t \mathcal{K}((w_t, I_t), \cdot)$, the predictor will be:

$$\begin{aligned}
\Lambda\alpha(x) &= \mathop{\mathbb{E}}_{(w,I)\sim p}\left[\alpha(w, I)\phi((w, I), x)\right] \\
&= \sum_{t=1}^{T} a_t \mathop{\mathbb{E}}_{(w,I)\sim p}\left[\mathcal{K}((w_t, I_t), (w, I))\phi_I(w, x_I)\right] \\
&= \sum_{J\subseteq[n]} \sum_{t:I_t=J} a_t \mathop{\mathbb{E}}_{(w,I)\sim p}\left[\mathcal{K}((w_t, I_t), (w, I))\phi_I(w, x_I)\right]. \qquad (19)
\end{aligned}$$

	Existing instantiation	STAR version
Parameter space	$\mathcal{W}$	$\mathcal{W} \times \mathcal{P}([n])$
Base predictor	$\phi(w, x) = \phi(\langle w, x\rangle)$	$\phi((w, I), x) = \phi(\langle w_I, x_I\rangle)$
Kernel	$\mathcal{K}_{\mathcal{W}}$	$\mathcal{K}((w_1, I_1),(w_2, I_2)) = \mathcal{K}_{\mathcal{W}}(w_1, w_2)\mathbb{1}[I_1 = I_2]$
Distribution	$p_{\mathcal{W}}$	$p(w, I) = p_{\mathcal{W}\|I}(w)p_{[n]}(I)$

Table 2: Cheat sheet for transforming an existing RKHS Weighting instantiation into a Structured Additive Regression model.

We've thus separated the predictor by feature subset, and so this is a STAR model $\Lambda\alpha(x) = \sum_{J\subseteq[n]} f_J(x_J)$, with:

$$f_J(x_J) := \sum_{t:I_t=J} a_t \mathop{\mathbb{E}}_{(w,I)\sim p}[\mathcal{K}((w_t, I_t),(w, I))\phi_I(w, x_I)]. \tag{20}$$

This framework for expressing an RKHS Weighting as a STAR model is in full generality. Of course, prior knowledge specific to a given problem can be inserted into the choices for a base predictor, kernel or distribution. For instance, we might be interested only in sequences of features when presented with text, time series, or other sequential data, in which case the distribution p can simply put a weight of 0 to any non-sequential feature subset. This family of STAR models is therefore highly flexible.

4.3 Transforming an existing instantiation into a STAR model

Perhaps the simplest way to instantiate an RKHS Weighting as a STAR model using the framework described in the previous section is to take an existing instantiation $(\mathcal{W}, \phi, \mathcal{K}_{\mathcal{W}}, p_{\mathcal{W}})$ which satisfies a few simple conditions:

1. $\mathcal{X} = \mathcal{W} = \mathbb{R}^n$.
2. The base predictor $\phi(w, x)$ is a function of $\langle w, x\rangle$, i.e. $\phi(w, x) = \phi(\langle w, x\rangle)$.
3. The distribution $p_{\mathcal{W}}$ on $\mathcal{W}$ generates each component independently.

Then, modifying the base predictor to work on feature subsets is almost trivially easy:

$$\phi((w, I), x) := \phi(\langle w_I, x_I\rangle). \tag{21}$$

In other words, simply ignore all the variables not in the feature subset. As for the distribution, the independence allows us to marginalize away the unwanted variables. The distribution $p(w, I)$ will be the product $p_{\mathcal{W}|I}(w)p_{[n]}(I)$, where $p_{[n]}$ is a chosen distribution over the feature subsets, and $p_{\mathcal{W}|I}(w)$ is the original distribution $p_{\mathcal{W}}$ marginalized to the features in I. Finally, we elect to use the indicator kernel $\mathcal{K}_{[n]}(I_1, I_2) := \mathbb{1}[I_1 = I_2]$ over the feature subsets, and use the kernel product $\mathcal{K}((w_1, I_1),(w_2, I_2)) := \mathcal{K}_{\mathcal{W}}(w_1, w_2)\mathcal{K}_{[n]}(I_1, I_2)$.[1] We end up with the new instantiation $(\mathcal{W} \times \mathcal{P}([n]), \phi, \mathcal{K}_{\mathcal{W}} \times \mathcal{K}_{[n]}, p)$. (Table 2 summarizes all of this.) Assuming a weight function $\alpha = \sum_{t=1}^{T} a_t\mathcal{K}((w_t, I_t), \cdot)$, the model will be:

$$\Lambda\alpha(x) = \sum_{J\subseteq[n]} \sum_{t:I_t=J} a_t p_{[n]}(I_t) \mathop{\mathbb{E}}_{w\sim p_{\mathcal{W}|I}} [\mathcal{K}_{\mathcal{W}}(w_t, w)\phi(\langle w, x_{I_t}\rangle)]. \tag{22}$$

The only element that still needs to be further specified is the distribution $p_{[n]}$ on the feature subsets. We list below the two most obvious candidates, but note that there is complete flexibility here for the user of the model to specify any distribution that they deem adequate for the dataset at hand.

[1] We use the indicator kernel for two related but distinct reasons. The first is the assumption that, in general, the comparison between parameters, i.e. the kernel evaluation $\mathcal{K}_{\mathcal{W}}(w_1, w_2)$, only makes sense if w_1 and w_2 are parameters over the same features. Second, it ensures that w_1 and w_2 are the same dimensionality, so that the kernel evaluation $\mathcal{K}_{\mathcal{W}}(w_1, w_2)$ is well-defined. There is some abuse of notation here, as we use the same symbol $\mathcal{K}_{\mathcal{W}}$ to denote a function over potentially different spaces (e.g. w_1 could be from $\mathbb{R}^3$, and w_2 from $\mathbb{R}^4$), but deem it benign enough, as it does simplify the expressions.

1. The simplest option is to set $p_{[n]}$ as the uniform distribution on feature subsets of size equal to k, where k becomes a hyperparameter of the algorithm. A slight variation of this is to use feature subsets of size up to k, allowing a larger variety of variable interactions.

2. Some datasets, like time series or genomic data, have structure, where adjacent features are correlated in some meaningful way. When presented with such sequential data, it is natural to consider sequences of features. We can then set $p_{[n]}$ as the uniform distribution over sequences of length k (or length up to k).

Because of the presence of the factor $p_{[n]}(I_t)$ in Equation (22), using the first option (arbitrary subsets of size k) with large k is not recommended. Indeed, since $p_{[n]}(I_t) = \frac{1}{\binom{n}{k}} \approx \frac{1}{n^k}$, the model output will tend to zero as k increases. Large coefficients a_t will be then required to offset the small density. This will lead to a large norm $\|\alpha\|_{\mathcal{H}}$ of the weight function, and therefore poorer guarantees (Dubé & Marchand, 2024). Sequential feature subsets do not suffer from this drawback. Since there are only $n - k + 1$ sequences of features of length k in an n-dimensional dataset, $p_{[n]}$ will always be of the order of $\frac{1}{n}$. When considering sequences of length up to k, then $p_{[n]}$ will be about $\frac{1}{kn}$, which is undeniably preferable over $\frac{1}{n^k}$.

We can use this transformation to immediately obtain STAR models based on instantiations RWSign and RWRelu, which can be learned using any algorithm from Dubé & Marchand (2024). (Note that instantiation RWStumps is already a GAM, so no transformation is required). This makes up a new family of STAR models, which are easy to train, as they do not require explicitly specifying feature interactions when using one of the learning algorithms from Dubé & Marchand (2024).

(For completeness, we describe in the appendix how to obtain the theoretical constants from Dubé & Marchand (2024) for a STAR RKHS model.)

5 Experiments

We introduced three new tools in this paper: STAR-SHAP, an algorithm for calculating the Shapley values of a STAR model; RWRelu, a new RKHS Weighting instantiation; and a new family of STAR models, obtained by transforming existing RKHS Weighting instantiations into STAR models. We present in the next few sections three experiments to test these new tools. First, Section 5.1 presents a simple regression performance experiment, designed to compare various regression algorithms to the new RWRelu and STAR RKHS Weighting models. The experiment of Section 5.2 investigates a scenario (time series regression) in which we expect the STAR RKHS Weightings to be better suited than the simple regression problems of the Section 5.1 experiment. Finally, we compare the computation time of STAR-SHAP to SHAP (Lundberg & Lee, 2017) in Section 5.3.

5.1 Regression performance

Three new models arise from Section 4: the RWRelu, and the STAR versions of RWSign and RWRelu. This calls for a simple regression experiment comparing the performance of these models to more bread and butter regression techniques, especially interpretable ones. Using 5-fold cross-validation to select hyperparameters, we trained a variety of models on several regression datasets. See Table 3 for information on the datasets, and the appendix for the detailed experimental setup.

We can look to the results in Table 4 to answer a few important questions.

Q1. How does RWRelu, the ReLU version of an RKHS Weighting, compare to RWSign and RWStumps, the two original instantiations, for regression tasks?

R1. We can see that RWRelu performs equally or better than RWSign on all but one dataset (Wine). On the other hand, RWStumps outperforms RWRelu on 4 of the 6 datasets. The STAR versions of RWRelu and RWSign outperform the other (for equal k) on some datasets, and not others, neither systematically standing out. These observations give some weight, but not much, to the conjecture that RWRelu is better suited to

	Training size	Test size	Dimensionality	Source (clickable links)
abalone	3132	1045	10	UCI
diabetes	331	111	10	Scikit-learn
housing	15480	5160	8	Scikit-learn
concrete	772	258	8	UCI
conductivity	15947	5316	81	UCI
wine	133	45	13	UCI

Table 3: Datasets of the regression experiment.

Table 4: Comparison of the regression performance of STAR RKHS Weightings and the ReLU instantiation, learned using Algorithm 3 of Dubé & Marchand (2024) (the Least Squares fit) with $T = 5000$, to a variety of other regression algorithms. k-STAR signifies the STAR version of the model using feature subsets of size k. The table values are the test R^2 (coefficient of determination). The highest (best) values for each dataset have been bolded.

Dataset Algorithm	abalone	diabetes	housing	concrete	conductivity	wine
DecisionTreeRegressor	0.499	0.105	0.677	0.796	0.882	0.867
EBM	0.552	0.331	**0.822**	**0.931**	**0.903**	0.897
KernelRidge	**0.587**	0.282	0.771	0.831	0.872	**0.951**
LinearRegression	0.543	0.359	0.591	0.623	0.736	0.804
RWRelu	0.577	0.349	0.734	0.835	0.842	0.832
RWRelu 1-STAR	0.555	0.314	0.662	0.780	0.777	0.808
RWRelu 2-STAR	0.577	0.305	0.731	0.847	0.805	0.843
RWRelu 3-STAR	0.580	0.332	0.744	0.847	0.568	0.845
RWRelu 4-STAR	0.579	0.350	0.737	0.846	0.032	0.835
RWRelu 5-STAR	0.578	0.334	0.729	0.844	0.001	0.837
RWRelu [1, 2, 3, 4, 5]-STAR	0.577	0.298	0.732	0.808	0.779	0.826
RWSign	0.578	0.348	0.711	0.828	0.728	0.893
RWSign 1-STAR	0.350	0.302	0.354	0.471	0.685	0.750
RWSign 2-STAR	0.579	0.369	0.696	0.835	0.816	0.868
RWSign 3-STAR	**0.586**	0.339	0.720	0.840	0.468	0.912
RWSign 4-STAR	0.585	0.349	0.708	0.794	0.003	0.896
RWSign 5-STAR	0.580	0.333	0.705	0.770	-0.000	0.899
RWSign [1, 2, 3, 4, 5]-STAR	0.580	**0.399**	0.698	0.826	0.694	0.840
RWStumps	0.567	0.339	0.768	0.911	0.859	0.864
SVR	0.564	0.293	0.770	0.799	0.862	**0.950**

regression than the other two RKHS Weighting instantiations. Rather, they might all be complementary, each with better and worse use cases.

Q2. How do the STAR versions of RKHS Weighting models perform in comparison to their original counterparts?

R2. There is a moderate loss of performance going from RWRelu to its 1-STAR version, a cost that might be acceptable to pay for the immense gain in interpretability. However, increasing k does improve the performance significantly on all but one dataset, Conductivity, and 3-STAR RWRelu is the best performing STAR version of the model, which shows that adding interactions can improve the quality of the model. The extremely poor performance of 3, 4 and 5-STAR models on Conductivity is caused by the higher dimensionality of the dataset. (See this footnote[2] for the in-depth explanation.) Indeed, larger feature subsets should not blindly be used except on low-dimensional datasets, or failure of learning might occur. As for the STAR versions of RWSign, they are less convincing than RWRelu, with the 1-STAR version in particular exhibiting extremely poor performance overall.

Q3. How does the new RWRelu instantiation perform compared to other regression algorithms?

R3. The two longstanding, bread and butter regression algorithms that are Kernel Ridge Regression and Support Vector Regression outperform RWRelu respectively on 4 of the 6, and 3 of the 6 datasets. This is far from an indictment of RWRelu.

Q4. Can interpretable RKHS Weightings perform similarly or better than other interpretable methods?

R4. The Explainable Boosting Machine (Lou et al., 2013) appears overall to be the best at both accuracy and interpretability, at least in this experiment. Kernel Ridge Regression is best on two datasets, and SVR, 3-STAR RWSign, and 1-to-5 STAR RWSign are each best on one dataset. Therefore, we can say that interpretable RKHS Weightings are at least sometimes competitive.

To conclude this experiment, we see that transforming an RKHS Weighting into a STAR model is a valid way of learning an interpretable model, at a relatively small cost of performance. However, other methods, especially Explainable Boosting Machines, often outperform these RKHS Weightings in both accuracy and interpretability.

In the next experiment, we will leverage prior knowledge to extract better performance out of the STAR RKHS Weightings.

5.2 Time series prediction performance

Perhaps the biggest takeaway from Table 4 is the excellence of the Explainable Boosting Machine (Lou et al., 2013) on both the accuracy and interpretability fronts. The RKHS Weightings did not shine particularly brightly. However, the most important characteristic of RKHS Weightings is their high flexibility in instantiating the model. We can improve performance by utilizing an instantiation better adapted to the problem at hand. In the specific case of STAR RKHS Weightings, a tool we can play with is the distribution on the feature subsets. For instance, instead of using the uniform distribution over all subsets of a certain length, we can use a uniform distribution over *sequences* of adjacent features (e.g. features $(2, 3, 4)$, or $(n-2, n-1, n)$). This will make the STAR RKHS instantiation much better suited to solve regression problems for which adjacent data features are correlated, such as time series. Therefore, a better comparison between STAR RKHS Weightings and the Explainable Boosting Machine would be on such datasets, using a distribution over sequences of features. This is what we do here.

Specifically, we run both the Explainable Boosting Machine (EBM) and STAR RKHS Weightings on time series datasets taken from the UCR Archive (Dau et al., 2018). Each dataset was cut into segments of length 11, so that each instance x has length 10, and the value to predict y is the 11-th value. See Table 5.

[2] We recall the presence of a factor $p_{[n]}(I_t)$ in Equation (22), the expression for a STAR RKHS Weighting. When using the uniform distribution over all possible feature subsets of size k, we have $p_{[n]}(I_t) = \frac{1}{\binom{n}{k}} \approx \frac{1}{n^k}$, which will be an extremely small value for high-dimensionality datasets, leading to vanishing outputs of the model. And indeed, the offending dataset, Conductivity, is far higher-dimensional than the other ones, proving the point.

	Training size	Test size	Dimensionality
ChlorineConcentration	6990	57585	10
Computers	16434	16434	10
ECG5000	6487	58487	10
FacesUCR	2388	24588	10
LargeKitchenAppliances	24684	24684	10
MelbournePedestrian	3579	7314	10

Table 5: Datasets of the time series prediction experiment.

Hyperparameters were selected by 5-fold cross-validation. See the appendix for the detailed experimental setup.

The results of this experiment can be found in Table 6. The RKHS Weightings fare much better in this experiment that the previous. Here, they (slightly) outperform the EBM on 5 of the 6 datasets. On the other hand, the EBM does systematically reach the highest training R^2, suggesting that some overfitting has taken place, even though the parameters were chosen by cross-validation. Perhaps a different set of candidate parameters would have led to a smaller generalization gap. Finally, we notice the expected trend that larger feature subsets on the STAR RKHS Weightings lead to better results, with the full RWRelu instantiation often being the best (although by a very small margin). Indeed, more complex interactions allow for greater expressivity of the model, so that it can better fit the data.

5.3 Computation time of the Shapley values

Although it is clear from the computational complexity that Algorithm 1 will be vastly more efficient than the brute force calculation of the Shapley values for STAR models, we can illustrate just how much through a simple experiment. At the same time, it will help us glean what a reasonable maximal number of variable interactions in a STAR model can be (i.e. the maximal size of the feature subsets).

Figure 1 shows the computation time of the Shapley values of STAR models using either STAR-SHAP (Algorithm 1) or SHAP (Lundberg & Lee, 2017). We notice, of course, the exponential growth of the computation time for SHAP, while STAR-SHAP cares little for the dimensionality when the size of the feature subsets is fixed. For completeness, we also show the computation time growth of using STAR-SHAP to calculate the Shapley values of a model which uses the full n-way interactions. We observe the expected exponential behavior, mirroring SHAP. (We note in passing that the comparison between the two algorithms is not quite fair, as SHAP is a well-optimized library, and STAR-SHAP has been implemented quite simply in Python. An optimized STAR-SHAP would have an even better showing.)

6 Limitations and future work

We list here a number of limitations to the work presented in this paper, and how each should be addressed in the future.

Algorithm 1 complexity. Algorithm 1 can theoretically calculate the Shapley values of any STAR model, but the computational cost is exponential in the size of the feature subsets. While this limits the size of the feature subsets that can be used in the model, this is likely inevitable, rather than a flaw of the algorithm. Still, as shown in Figure 1, fairly complex models, with up to around 8-way interactions, are usable.

Instantiating RKHS Weightings. The main weakness of RKHS Weightings remains the requirement of solving a complex integral for any new instantiation. We have introduced one new instantiation, but it is still an open problem to figure out how to access the near limitless potential of this family of models. Indeed, it is almost certain that better results than those found in Tables 4 and 6 can be obtained by using different

Table 6: Comparison of the time series prediction performance of STAR RKHS Weightings and the ReLU instantiation, learned using Algorithm 3 of Dubé & Marchand (2024) (the Least Squares fit) with $T = 5000$, to the Explainable Boosting Machine (EBM). k-STAR signifies the STAR version of the model using feature subsets of size k. The highest (best) values for each dataset have been bolded.

Dataset	Algorithm	Training R^2	Test R^2	Training time (s)
ChlorineConcentration	EBM	**0.955**	0.916	63.853
	RWRelu	0.934	**0.928**	3.307
	RWRelu 1-STAR	0.638	0.642	36.375
	RWRelu 2-STAR	0.813	0.805	40.390
	RWRelu 3-STAR	0.907	0.899	44.500
	RWRelu 4-STAR	0.928	0.920	48.720
	RWRelu 5-STAR	0.939	**0.927**	54.159
	RWRelu [1, 2, 3, 4, 5]-STAR	0.924	0.906	19.823
Computers	EBM	**0.882**	0.842	16.291
	RWRelu	0.821	**0.851**	6.085
	RWRelu 1-STAR	0.786	0.838	44.742
	RWRelu 2-STAR	0.802	0.840	50.501
	RWRelu 3-STAR	0.811	0.848	55.049
	RWRelu 4-STAR	0.814	0.847	58.744
	RWRelu 5-STAR	0.818	0.848	64.858
	RWRelu [1, 2, 3, 4, 5]-STAR	0.817	**0.849**	29.086
ECG5000	EBM	**0.943**	**0.886**	23.525
	RWRelu	0.900	0.882	3.144
	RWRelu 1-STAR	0.843	0.841	36.361
	RWRelu 2-STAR	0.877	0.871	40.024
	RWRelu 3-STAR	0.883	0.875	43.769
	RWRelu 4-STAR	0.893	0.879	48.122
	RWRelu 5-STAR	0.899	0.882	53.384
	RWRelu [1, 2, 3, 4, 5]-STAR	0.894	0.882	19.157
FacesUCR	EBM	**0.868**	0.661	7.674
	RWRelu	0.715	**0.704**	1.929
	RWRelu 1-STAR	0.679	0.698	33.011
	RWRelu 2-STAR	0.690	0.702	35.613
	RWRelu 3-STAR	0.699	**0.705**	38.827
	RWRelu 4-STAR	0.700	**0.705**	42.990
	RWRelu 5-STAR	0.709	**0.706**	49.571
	RWRelu [1, 2, 3, 4, 5]-STAR	0.704	**0.706**	15.570
LargeKitchenAppliances	EBM	**0.840**	0.743	19.103
	RWRelu	0.813	0.743	8.775
	RWRelu 1-STAR	0.764	0.724	52.104
	RWRelu 2-STAR	0.786	**0.748**	56.938
	RWRelu 3-STAR	0.777	0.736	60.735
	RWRelu 4-STAR	0.803	0.746	68.894
	RWRelu 5-STAR	0.810	0.745	75.208
	RWRelu [1, 2, 3, 4, 5]-STAR	0.812	**0.749**	37.741
MelbournePedestrian	EBM	**0.979**	**0.924**	9.731
	RWRelu	0.946	**0.925**	2.335
	RWRelu 1-STAR	0.888	0.888	35.930
	RWRelu 2-STAR	0.921	0.902	36.519
	RWRelu 3-STAR	0.943	**0.923**	40.523
	RWRelu 4-STAR	0.943	**0.925**	44.806
	RWRelu 5-STAR	0.943	**0.924**	50.667
	RWRelu [1, 2, 3, 4, 5]-STAR	0.941	0.919	16.232

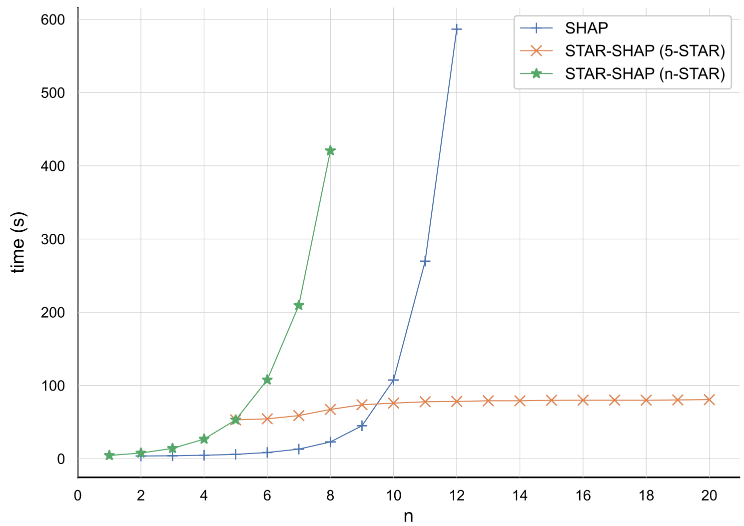

Figure 1: Computation time of the Shapley values of a Structured Additive Regression model with respect to the dimensionality of the dataset. For every value of n, a STAR RKHS Weighting model was trained on a random synthetic dataset of 100 examples. Then, SHAP and STAR-SHAP (Algorithm 1) were used to calculate the Shapley values of all examples. 5-STAR indicates that the STAR model had size 5 feature subsets (5-way interactions). Similarly, n-STAR models used all n possible interactions.

instantiations. As Dubé & Marchand (2024) suggested, Monte Carlo approximation of the expectations might provide a good enough solution, and warrants an in-depth look.

Generic STAR RKHS Weightings. As seen in Table 4, the most generic version of STAR RKHS Weightings, i.e. using the uniform distribution on all possible feature subsets of size k, have somewhat limited use, since their output scales as $\frac{1}{\binom{n}{k}}$ (n being the dimensionality of the dataset). As we've already addressed, the best option is to insert prior knowledge into the model, as we have done with time series datasets by considering sequences of variables. Extending this principle to other types of structured data would increase the applicability of STAR RKHS Weighting models.

7 Conclusion

In this paper, we've derived an efficient algorithm, named STAR-SHAP, for calculating the Shapley values of any Structured Additive Regression (STAR) model. We've introduced a new RKHS Weightings instantiation, and showed how to obtain RKHS Weightings that are STAR models, giving rise to a new family of STAR models, all of which are now interpretable thanks to STAR-SHAP. We've tested the prediction performance of the introduced models. While the STAR RKHS Weightings did not rise to the level of state of the art interpretable algorithms on generic regression datasets, they proved quite capable in the context of time series prediction when infused with adequate prior knowledge. Further work to increase the breadth of usable RKHS Weighting instantiations is warranted, as well as work to expand the types of datasets where RKHS Weightings can perform well.

References

Sebastian Bordt and Ulrike von Luxburg. From shapley values to generalized additive models and back. In *International Conference on Artificial Intelligence and Statistics*, pp. 709–745. PMLR, 2023.

Eugenio Brusa, Luca Cibrario, Cristiana Delprete, and Luigi Gianpio Di Maggio. Explainable ai for machine fault diagnosis: understanding features' contribution in machine learning models for industrial condition monitoring. *Applied Sciences*, 13(4):2038, 2023.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1721–1730, 2015.

Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

Gabriel Dubé and Mario Marchand. RKHS Weightings of Functions. working paper or preprint, July 2024. URL `https://hal.science/hal-04236058`.

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

Daniel Vidali Fryer, Inga Strümke, and Hien Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. *IEEE Access*, PP:1–1, 2021. URL `https://api.semanticscholar.org/CorpusID:231985458`.

Trevor J Hastie. Generalized additive models. In *Statistical models in S*, pp. 249–307. Routledge, 2017.

Giles Hooker. Discovering additive structure in black box functions. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 575–580, 2004.

Xuanxiang Huang and Joao Marques-Silva. The inadequacy of shapley values for explainability, 2023.

F Kuo, I Sloan, Grzegorz Wasilkowski, and Henryk Woźniakowski. On decompositions of multivariate functions. *Mathematics of computation*, 79(270):953–966, 2010.

Gabriel Laberge, Yann Pequignot, Alexandre Mathieu, Foutse Khomh, and Mario Marchand. Partial order in chaos: Consensus on feature attributions in the rashomon set. *Journal of Machine Learning Research*, 24(364):1–50, 2023.

Gabriel Laberge, Yann Pequignot, Mario Marchand, and Foutse Khomh. Tackling the XAI Disagreement Problem with Regional Explanations. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238, Valencia, Spain, May 2024. URL `https://hal.science/hal-04480870`.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 623–631, 2013.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

Michael Mayer, Steven C Bourassa, Martin Hoesli, and Donato Scognamiglio. Structured additive regression and tree boosting. *Swiss Finance Institute Research Paper*, (21-83), 2021.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018.

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*, pp. 1135–1144, 2016.

Cynthia Rudin, Chudi Zhong, Lesia Semenova, Margo Seltzer, Ronald Parr, Jiachang Liu, Srikar Katta, Jon Donnelly, Harry Chen, and Zachery Boner. Amazing things come from having many good models. 2024.

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

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

Eyal Winter. The shapley value. *Handbook of game theory with economic applications*, 3:2025–2054, 2002.

Liang Zhang and Zhelun Chen. Large language model-based interpretable machine learning control in building energy systems. *arXiv preprint arXiv:2402.09584*, 2024.

A Proofs

First, we quickly prove Equation (9).

of Equation (9). We have:

$$
\begin{aligned}
\phi_i^{\mathrm{SHAP}}(h,x) &:= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[h(r_{\pi_{:i}\cup\{i\}}(z,x)) - h(r_{\pi_{:i}}(z,x))\right] \\
&= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[\left(\sum_{j\in\pi_{:i}\cup\{i\}} f_j(x_j) + \sum_{j\notin\pi_{:i}\cup\{i\}} f_j(z_j)\right) - \left(\sum_{j\in\pi_{:i}} f_j(x_j) - \sum_{j\notin\pi_{:i}} f_j(z_j)\right)\right] \\
&= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[\left(\sum_{j\in\pi_{:i}\cup\{i\}} f_j(x_j) - \sum_{j\in\pi_{:i}} f_j(x_j)\right) + \left(\sum_{j\notin\pi_{:i}\cup\{i\}} f_j(z_j) - \sum_{j\notin\pi_{:i}} f_j(z_j)\right)\right] \\
&= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[\left(f_i(x_i)\right) - \left(\sum_{j\in\pi_{:i}\cup\{i\}} f_j(z_j) - \sum_{j\in\pi_{:i}\cup\{i\}} f_j(z_j) + f_i(z_i)\right)\right] \\
&= f_i(x_i) - \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[f_i(z_i)\right] \\
&= f_i(x_i) - \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \left[f_i(z_i)\right].
\end{aligned}
$$

□

The main result to prove is Theorem 3.1, which follows from two lemmas further below.

of Theorem 3.1. We are calculating the Shapley values of a STAR model $h(x) = \sum_{I\subseteq[n]} f_I(x_I)$, which can be written as:

$$
\begin{aligned}
\phi_i^{\mathrm{SHAP}}(h,x) &:= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[h(r_{\pi_{:i}\cup\{i\}}(z,x)) - h(r_{\pi_{:i}}(z,x))\right] \\
&= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[\sum_{I\subseteq[n]} f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I) - \sum_{I\subseteq[n]} f_I(r_{\pi_{:i}}(z,x)_I)\right] \\
&= \mathop{\mathbb{E}}_{z\sim\mathcal{S}} \left[\sum_{I\subseteq[n]} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)} \left[f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I) - f_I(r_{\pi_{:i}}(z,x)_I)\right]\right].
\end{aligned}
$$

We can immediately cut out most of the computations by noticing two facts:

1. Most functions f_I are 0.

2. If $i \notin I$, then $r_{\pi_{:i}\cup\{i\}}(z,x)_I = r_{\pi_{:i}}(z,x)_I$. Hence, the entire expectation:

$$E_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I) - f_I(r_{\pi_{:i}}(z,x)_I)\right] \tag{23}$$

   is 0.

The Shapley values therefore boil down to the following formula:

$$\phi_i^{\text{SHAP}}(h,x) = \mathop{\mathbb{E}}_{z\sim\mathcal{S}}\left[\sum_{I\subseteq[n]:f_I\neq 0, i\in I} \mathop{\mathbb{E}}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)\right] - \mathop{\mathbb{E}}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}}(z,x)_I)\right]\right]. \tag{24}$$

Lemma A.1 gives the value of $\mathbb{E}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)\right]$, and Lemma A.2 gives the value of $\mathbb{E}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}}(z,x)_I)\right]$. ☐

Lemma A.1. *Consider the definitions of Section 2, assuming that $i \in I$, and the following:*

$$\mathcal{A}^+(i,I) := \{A \subset I | i \in A, 1 \leq |A| < |I|\}. \tag{25}$$

In other words, $\mathcal{A}^+(i,I)$ is the family of strict subsets of I which contain i. Then we have:

$$\mathop{\mathbb{E}}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)\right] = \frac{f_I(x_I)}{|I|} + \sum_{A\in\mathcal{A}^+(i,I)} \frac{f_I(r_A(z,x)_I)}{|A|\binom{|I|}{|A|}}. \tag{26}$$

Lemma A.2. *Consider the definitions of Section 2, assuming that $i \in I$, as well as:*

$$\mathcal{A}^-(i,I) := \{A \subseteq I | i \in A, 1 < |A| \leq |I|\}. \tag{27}$$

In other words, $\mathcal{A}^-(i,I)$ is the family of subsets of I which contain i and at least one other element. Then we have:

$$\mathop{\mathbb{E}}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}}(z,x)_I)\right] = \frac{f_I(z_I)}{|I|} + \sum_{A\in\mathcal{A}^-(i,I)} \frac{f_I(r_{A\setminus\{i\}}(z,x)_I)}{|A|\binom{|I|}{|A|}}. \tag{28}$$

of Lemma A.1. We can separate the expectation into two parts:

$$\mathop{\mathbb{E}}_{\pi\sim U(\Omega)}\left[f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)\right] = \underbrace{\frac{1}{n!}\sum_{\substack{\pi\in\Omega\\ I\subseteq\pi_{:i}\cup\{i\}}} f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)}_{\text{Term 1}} + \underbrace{\frac{1}{n!}\sum_{\substack{\pi\in\Omega\\ I\setminus(\pi_{:i}\cup\{i\})\neq\emptyset}} f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)}_{\text{Term 2}}. \tag{29}$$

Calculating Term 1. The key to calculating the first half of the right-hand side of the previous equation, which we'll refer to as Term 1, is to notice that $r_{\pi_{:i}\cup\{i\}}(z,x)_I = x_I$ for all the permutations π that satisfy $I \subseteq \pi_{:i} \cup \{i\}$, and so the value $f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I)$ will be constant over all those permutations. We'll simply have $f_I(r_{\pi_{:i}\cup\{i\}}(z,x)_I) = f_I(x_I)$. All that we need to do is calculate the number of such permutations. There is an elegant combinatorics argument that gives us the solution immediately.

Consider any permutation π of $[n]$. This permutation satisfies the condition $I \subseteq \pi_{:i} \cup \{i\}$ if and only if i is to the right of every other variable of I. This rightmost position is one possibility of out of $|I|$. This means that exactly 1 out of every $|I|$ permutations satisfies the condition. The number of permutations we seek is $\frac{n!}{|I|}$. We therefore have:

$$\text{Term 1} = \frac{1}{n!}\frac{n!}{|I|}f_I(x_I) = \frac{f_I(x_I)}{|I|}. \tag{30}$$

Calculating Term 2. Here, we consider all the permutations π such that $I \setminus (\pi_{:i} \cup \{i\}) \neq \emptyset$, i.e. the feature subset I contains variables not in $\pi_{:i} \cup \{i\}$. These variables will be replaced when calculating the function. Defining $A := I \cap (\pi_{:i} \cup \{i\})$, the set of shared variables (note that we always have $i \in A$), we'll have:

$$r_{\pi_{:i} \cup \{i\}}(z, x)_I = r_A(z, x)_I.$$

The function $f_I(r_{\pi_{:i} \cup \{i\}}(z, x)_I)$ will therefore take a different value for each possible A, of which there are:

$$\begin{aligned} \left|\mathcal{A}^+(i, I)\right| &= \sum_{k=0}^{|I|-2} \binom{|I|-1}{k} \\ &= \sum_{k=0}^{|I|-1} \binom{|I|-1}{k} - \binom{|I|-1}{|I|-1} \\ &= 2^{|I|-1} - 1. \end{aligned} \tag{31}$$

(We can choose 0 up to $|I| - 2$ variables to accompany i in making up $A \in \mathcal{A}^+(i, I)$.) For a given set $A \in \mathcal{A}^+(i, I)$ of shared variables, we must calculate the number of permutations π that are such that $A = I \cap (\pi_{:i} \cup \{i\})$. Call that number $N(n, |I|, |A|)$. We'll have:

$$\text{Term 2} = \sum_{A \in \mathcal{A}^+(i,I)} \frac{N(n, |I|, |A|)}{n!} f_I(r_A(z, x)_I). \tag{32}$$

Let's now calculate $N(n, |I|, |A|)$.

1. First, choose the $|I|$ positions for the variables in I. There are $\binom{n}{|I|}$ possibilities.
2. Of these $|I|$ positions, there is only one choice for i itself, since the variables in $A \setminus \{i\}$ must be to the left of it, and the variables in $I \setminus A$ to the right.
3. We can permute the $|A| - 1$ variables of $A \setminus \{i\}$, the $|I| - |A|$ variables of $I \setminus A$, and the $n - |I|$ variables in $[n] \setminus I$, for a total of $(|A| - 1)!(|I| - |A|)!(n - |I|)!$ permutations.

Assembling these facts gives us the formula:

$$N(n, |I|, |A|) := (|A| - 1)!(|I| - |A|)!(n - |I|)! \binom{n}{|I|} \tag{33}$$

We can simplify this further by expanding the binomial coefficient:

$$\begin{aligned} N(n, |I|, |A|) &= (|A| - 1)!(|I| - |A|)!(n - |I|)! \binom{n}{|I|} \\ &= \frac{n!(|A| - 1)!(|I| - |A|)!(n - |I|)!}{|I|!(n - |I|)!} \\ &= \frac{n!}{|A|\binom{|I|}{|A|}}. \end{aligned} \tag{34}$$

This gives us the result. $\square$

of Lemma A.2. The proof is quite similar to that of Lemma A.1. We can separate the expectation into two parts:

$$\mathop{\mathbb{E}}_{\pi \sim U(\Omega)} [f_I(r_{\pi_{:i}}(z, x)_I)] = \underbrace{\frac{1}{n!} \sum_{\substack{\pi \in \Omega \\ I \cap \pi_{:i} = \emptyset}} f_I(r_{\pi_{:i}}(z, x)_I)}_{\text{Term 1}} + \underbrace{\frac{1}{n!} \sum_{\substack{\pi \in \Omega \\ I \cap \pi_{:i} \neq \emptyset}} f_I(r_{\pi_{:i}}(z, x)_I)}_{\text{Term 2}}. \tag{35}$$

Calculating Term 1. Notice that $r_{\pi_{:i}}(z,x)_I = z_I$ for all the permutations π that satisfy $I \cap \pi_{:i} = \emptyset$, and so the value $f_I(r_{\pi_{:i}}(z,x)_I)$ will be constant over all those permutations. We'll simply have $f_I(r_{\pi_{:i}}(z,x)_I) = f_I(z_I)$. All that we need to do to calculate the number of such permutations. However, a simple symmetry argument gives us the answer. In the proof of Lemma A.1, we showed that the number of permutations π such that $I \subseteq \pi_{:i} \cup \{i\}$ is $\frac{n!}{|I|}$. In fact, we can notice that this number should be the same as the number of permutations that satisfy $I \cap \pi_{:i} = \emptyset$. Indeed, in the first case, all variables of $I \setminus \{i\}$ must be to the left of i in $\pi_{:i}$. In the second, they must be to the right. There are exactly as many permutations that satisfy each condition.

Calculating Term 2. Here, we consider all the permutations π such that $I \cap \pi_{:i} \neq \emptyset$, i.e. the feature subset I shares at least one variable with $\pi_{:i}$. In this situation, some or all variables are replaced when calculating the function. Using the same definition $A := I \cap (\pi_{:i} \cup \{i\})$ as in the proof of Lemma A.1, we'll have:

$$r_{\pi_{:i}}(z,x)_I = r_{A\setminus\{i\}}(z,x)_I.$$

The function $f_I(r_{\pi_{:i}}(z,x)_I)$ will therefore take a different value for each possible A. Since each A must contain at least two elements (i, and one more variable so that $I \cap \pi_{:i} \neq \emptyset$), and also $A = I$ is now acceptable, the intersection A is taken from the set $\mathcal{A}^-(i,I)$, which has the same cardinality as the set $\mathcal{A}^+(i,I)$, namely $2^{|I|-1} - 1$. We also know from the previous proof that there are $N(n,|I|,|A|) = \frac{n!}{|A|\binom{|I|}{|A|}}$ permutations π such that $A = I \cap (\pi_{:i} \cup \{i\})$. We therefore have:

$$\text{Term 2} = \sum_{A \in \mathcal{A}^-(i,I)} \frac{f_I(r_{A\setminus\{i\}}(z,x)_I)}{|A|\binom{|I|}{|A|}}, \tag{36}$$

which concludes the proof. □

B Calculus for RWRelu expectation

Here we calculate the expectation $\mathbb{E}_{w\sim p}\left[\mathcal{K}(u,w)\phi(w,x)\right]$ for RWRelu (required to calculate the output of the model) through a series of lemmas. First, we recall a pair of lemmas from Dubé & Marchand (2024).

Lemma B.1 (Lemma 15 of Dubé & Marchand (2024))**.** *Consider a Hilbert space $\mathcal{W}$. Let $u, w \in \mathcal{W}$ and $a, b > 0$. Then:*

$$\frac{\|w-u\|^2}{a} + \frac{\|w\|^2}{b} = \left(\frac{1}{a} + \frac{1}{b}\right)\left\|w - \frac{1}{1+\frac{a}{b}}u\right\|^2 + \frac{1}{a+b}\|u\|^2.$$

Lemma B.2 (Lemma 16 of Dubé & Marchand (2024))**.** *We have:*

$$\int_{\mathbb{R}^n} e^{-a\|w-u\|^2}\mathrm{d}w = \left(\frac{\pi}{a}\right)^{\frac{n}{2}}.$$

Lemma B.3. *We have:*

$$\int_{-\infty}^{\infty} e^{-(w-u)^2/2\gamma^2}\max(0,wx)\mathrm{d}w = \frac{\gamma|x|}{\sqrt{2}}\left(\sqrt{2}\gamma e^{-u^2/2\gamma^2} + \sqrt{\pi}u\left(\operatorname{sign}(x) + \operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right)\right)\right).$$

Proof. We have:

$$\begin{aligned}\int_{-\infty}^{\infty} e^{-(w-u)^2/2\gamma^2}\max(0,wx)\mathrm{d}w &= \int_{-\infty}^{\infty} e^{-t^2}\max\left(0,\left(\sqrt{2}\gamma t+u\right)x\right)\sqrt{2}\gamma\mathrm{d}w \qquad \left(t := \tfrac{w-u}{\sqrt{2}\gamma},\ \mathrm{d}t = \tfrac{\mathrm{d}w}{\sqrt{2}\gamma}\right)\\ &= \sqrt{2}\gamma\int_{-\infty}^{\infty} e^{-t^2}\max\left(0,\left(\sqrt{2}\gamma t+u\right)x\right)\mathrm{d}t.\end{aligned}$$

Case 1. If $x \geq 0$, we have:

$$\begin{aligned}
\sqrt{2}\gamma \int_{-\infty}^{\infty} e^{-t^2} \max\Big(0, \Big(\sqrt{2}\gamma t + u\Big)x\Big)\mathrm{d}t \\
&= \sqrt{2}\gamma \int_{-u/\sqrt{2}\gamma}^{\infty} e^{-t^2} \Big(\sqrt{2}\gamma t + u\Big)x\mathrm{d}t \\
&= \sqrt{2}\gamma x\bigg(\sqrt{2}\gamma \int_{-u/\sqrt{2}\gamma}^{\infty} te^{-t^2}\mathrm{d}t + u\int_{-u/\sqrt{2}\gamma}^{\infty} e^{-t^2}\mathrm{d}t\bigg) \\
&= \sqrt{2}\gamma x\left(\sqrt{2}\gamma\left(\frac{-e^{-t^2}}{2}\bigg|_{t=-u/\sqrt{2}\gamma}^{t=\infty}\right) + u\left[\int_{-u/\sqrt{2}\gamma}^{0} e^{-t^2}\mathrm{d}t + \int_{0}^{\infty} e^{-t^2}\mathrm{d}t\right]\right) \\
&= \sqrt{2}\gamma x\bigg(\frac{\gamma}{\sqrt{2}}e^{-u^2/2\gamma^2} + u\bigg[\frac{\sqrt{\pi}}{2}\operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right) + \frac{\sqrt{\pi}}{2}\operatorname{erf}(\infty)\bigg]\bigg) \\
&= \frac{\gamma x}{\sqrt{2}}\bigg(\sqrt{2}\gamma e^{-u^2/2\gamma^2} + \sqrt{\pi}u\bigg[1 + \operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right)\bigg]\bigg).
\end{aligned}$$

Case 2. If $x < 0$, we have instead:

$$\begin{aligned}
\sqrt{2}\gamma \int_{-\infty}^{\infty} e^{-t^2} \max\Big(0, \Big(\sqrt{2}\gamma t + u\Big)x\Big)\mathrm{d}t \\
&= \sqrt{2}\gamma \int_{-\infty}^{-u/\sqrt{2}\gamma} e^{-t^2} \Big(\sqrt{2}\gamma t + u\Big)x\mathrm{d}t \\
&= \sqrt{2}\gamma x\bigg(\sqrt{2}\gamma \int_{-\infty}^{-u/\sqrt{2}\gamma} te^{-t^2}\mathrm{d}t + u\int_{-\infty}^{-u/\sqrt{2}\gamma} e^{-t^2}\mathrm{d}t\bigg) \\
&= \sqrt{2}\gamma x\left(\sqrt{2}\gamma\left(\frac{-e^{-t^2}}{2}\bigg|_{t=-\infty}^{t=-u/\sqrt{2}\gamma}\right) + u\left[\int_{-\infty}^{0} e^{-t^2}\mathrm{d}t - \int_{-u/\sqrt{2}\gamma}^{0} e^{-t^2}\mathrm{d}t\right]\right) \\
&= \sqrt{2}\gamma x\bigg(\frac{-\gamma}{\sqrt{2}}e^{-u^2/2\gamma^2} + u\bigg[\frac{\sqrt{\pi}}{2}\operatorname{erf}(\infty) - \frac{\sqrt{\pi}}{2}\operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right)\bigg]\bigg) \\
&= \frac{\gamma x}{\sqrt{2}}\bigg(-\sqrt{2}\gamma e^{-u^2/2\gamma^2} + \sqrt{\pi}u\bigg[1 - \operatorname{erf}\left(\frac{u}{\sqrt{2}\gamma}\right)\bigg]\bigg).
\end{aligned}$$

Both cases can be combined using the sign of x, leading to the desired result. ☐

Lemma B.4. *We have*

$$\begin{aligned}
\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2} \max(0, \langle w, x\rangle)\mathrm{d}w \\
&= \Big(\sqrt{2\pi\gamma^2}\Big)^{n-1}\left[\frac{\gamma\|x\|}{\sqrt{2}}\left(\sqrt{2}\gamma e^{-\frac{\langle u,x\rangle^2}{2\gamma^2\|x\|^2}} + \sqrt{\pi}\frac{\langle u, x\rangle}{\|x\|}\left[1 + \operatorname{erf}\left(\frac{\langle u, x\rangle}{\sqrt{2}\gamma\|x\|}\right)\right]\right)\right].
\end{aligned}$$

Proof. Calculate the integral using an orthonormal basis $\{v_1, \ldots, v_n\}$ of $\mathbb{R}^n$ such that $v_n := \frac{x}{\|x\|}$. Write $w = (w_1, \ldots, w_n)$ in this new basis (i.e. $w_i := \langle w, v_i\rangle$ for all i), and similarly $(u_1, \ldots, u_n)$ for u. Under this change of coordinates, the integral becomes:

$$\begin{aligned}
\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2} \max(0, \langle w, x\rangle)\mathrm{d}w \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}^{n-1}} e^{-\left[\sum_{i=1}^{n-1}(w_i-u_i)^2 + (w_n-u_n)^2\right]/2\gamma^2} \max(0, u_n\|x\|)\mathrm{d}w_1 \ldots \mathrm{d}w_n.
\end{aligned}$$

We are left with a product of n independent integrals:

$$\begin{aligned}
&\int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2} \max(0, \langle w, x\rangle)\mathrm{d}w \\
&\qquad = \int_{\mathbb{R}^{n-1}} e^{-\sum_{i=1}^{n-1}(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_1 \dots \mathrm{d}w_{n-1} \int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2} \max(0, w_n\|x\|)\mathrm{d}w_n \\
&\qquad = \int_{\mathbb{R}^{n-1}} \prod_{i=1}^{n-1} e^{-(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_1 \dots \mathrm{d}w_{n-1} \int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2} \max(0, w_n\|x\|)\mathrm{d}w_n \\
&\qquad = \prod_{i=1}^{n-1} \int_{-\infty}^{\infty} e^{-(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_i \int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2} \max(0, w_n\|x\|)\mathrm{d}w_n.
\end{aligned}$$

For each i, Lemma B.2 gives us:

$$\int_{\mathbb{R}} e^{-(w_i-u_i)^2/2\gamma^2}\mathrm{d}w_i = \sqrt{2\pi\gamma^2}.$$

Also, Lemma B.3 gives us:

$$\begin{aligned}
&\int_{\mathbb{R}} e^{-(w_n-u_n)^2/2\gamma^2} \max(0, w_n\|x\|)\mathrm{d}w_n \\
&\qquad = \frac{\gamma\|x\|}{\sqrt{2}}\left(\sqrt{2}\gamma e^{-u_n^2/2\gamma^2} + \sqrt{\pi}u_n\left(\operatorname{sign}(\|x\|) + \operatorname{erf}\left(\frac{u_n}{\sqrt{2}\gamma}\right)\right)\right).
\end{aligned}$$

Finally, since $u_n = \frac{\langle u,x\rangle}{\|x\|}$ and $\operatorname{sign}(\|x\|)$, we have the result. ☐

of Theorem 4.1. We have:

$$\begin{aligned}
\mathbb{E}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \int_{\mathbb{R}^n} e^{-\|w-u\|^2/2\gamma^2} e^{-\|w\|^2/2\sigma^2} \max\left(0, \langle w, x\rangle\right)\mathrm{d}w \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}} \int_{\mathbb{R}^n} e^{-\left(\frac{1}{2\gamma^2}+\frac{1}{2\sigma^2}\right)\left\|w-\frac{u}{1+\frac{\gamma^2}{\sigma^2}}\right\|^2} \max(0, \langle w, x\rangle)\mathrm{d}w \quad \text{(Lemma B.1)} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}} \int_{\mathbb{R}^n} e^{-\left\|w-u'\right\|^2/2\zeta^2} \max(0, \langle w, x\rangle)\mathrm{d}w.
\end{aligned}$$

Applying Lemma B.4, we obtain:

$$\begin{aligned}
&\mathbb{E}_{w\sim p}[\mathcal{K}(u,w)\phi(w,x)] \\
&\quad = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}} \left(\sqrt{2\pi\zeta^2}\right)^{n-1} \left[\frac{\zeta\|x\|}{\sqrt{2}}\left(\sqrt{2}\zeta e^{-\frac{\langle u',x\rangle^2}{2\zeta^2\|x\|^2}} + \sqrt{\pi}\frac{\langle u',x\rangle}{\|x\|}\left[1+\operatorname{erf}\left(\frac{\langle u',x\rangle}{\sqrt{2}\zeta\|x\|}\right)\right]\right)\right] \\
&\quad = \left(1+\frac{\sigma^2}{\gamma^2}\right)^{-n/2} e^{\frac{-\|u\|_2^2}{2\sigma^2+2\gamma^2}} \frac{\|x\|}{2\sqrt{\pi}}\left(\sqrt{2}\zeta e^{-\frac{\langle u',x\rangle^2}{2\zeta^2\|x\|^2}} + \sqrt{\pi}\frac{\langle u',x\rangle}{\|x\|}\left[1+\operatorname{erf}\left(\frac{\langle u',x\rangle}{\sqrt{2}\zeta\|x\|}\right)\right]\right).
\end{aligned}$$

☐

C Theoretical constants of RWRelu

For each of the two instantiations Dubé & Marchand (2024) introduce, they provide the value of two theoretical constants, κ and θ, relevant for their theoretical guarantees. For a given instantiation $(\mathcal{W}, \phi, \mathcal{K}, p)$, we have:

$$\kappa := \sup_{x\in\mathcal{X}} \sqrt{\mathbb{E}_{w\sim p}\left[\|\phi(w,x)\mathcal{K}(w,\cdot)\|_{\mathcal{H}}^2\right]} = \sup_{x\in\mathcal{X}} \sqrt{\mathbb{E}_{w\sim p}\left[\mathcal{K}(w,w)\phi(w,x)^2\right]} \tag{37}$$

$$\theta := \sup_{x\in\mathcal{X}} \left\| \mathop{\mathbb{E}}_{w\sim p}\left[\phi(w,x)\mathcal{K}(w,\cdot)\right] \right\|_{\mathcal{H}} = \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\mathop{\mathbb{E}}_{u\sim p}\left[\mathcal{K}(u,w)\phi(u,x)\phi(w,x)\right]}. \tag{38}$$

For completeness, we give here the exact value of κ and an upper bound for θ, and the relevant calculus, in the case of instantiation RWRelu. We begin by a couple of lemmas to help with the calculus.

Lemma C.1. *Consider $\sigma > 0$ and $\gamma > 0$. Then:*

$$\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\|u-w\|^2/2\gamma^2} e^{-\langle u,w\rangle/\sigma^2}\mathrm{d}u\mathrm{d}w = (2\pi)^n\left(\frac{\gamma^2\sigma^4}{2\sigma^2-\gamma^2}\right)^{n/2}. \tag{39}$$

Proof.

$$\begin{aligned}
\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}}\mathrm{d}u\mathrm{d}w &= \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|t\|^2}{2\gamma^2}} e^{-\frac{\langle t+w,w\rangle}{\sigma^2}}\mathrm{d}t\mathrm{d}w \qquad (t := u-w,\ \mathrm{d}t = \mathrm{d}u)\\
&= \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|t\|^2}{2\gamma^2}} e^{-\frac{\|w\|^2}{\sigma^2}} e^{-\frac{\langle t,w\rangle}{\sigma^2}}\mathrm{d}t\mathrm{d}w\\
&= \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}}\left[\int_{\mathbb{R}^n} e^{-\frac{\|t\|^2}{2\gamma^2}} e^{-\frac{\langle t,w\rangle}{\sigma^2}}\mathrm{d}t\right]\mathrm{d}w\\
&= \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}}\left[\int_{\mathbb{R}^n} e^{-\frac{\left\|t+\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}} e^{\frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\mathrm{d}t\right]\mathrm{d}w\\
&= \int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}} e^{\frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\left[\int_{\mathbb{R}^n} e^{-\frac{\left\|t+\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\mathrm{d}t\right]\mathrm{d}w\\
&= \left(\sqrt{2\pi\gamma^2}\right)^n\int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{\sigma^2}} e^{\frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2}}\mathrm{d}w.
\end{aligned}$$

Then, simplifying the exponent:

$$\begin{aligned}
-\frac{\|w\|^2}{\sigma^2} + \frac{\left\|\frac{\gamma^2 w}{\sigma^2}\right\|^2}{2\gamma^2} &= -\frac{\|w\|^2}{2\sigma^2}\left(2-\frac{\gamma^2}{\sigma^2}\right)\\
&= -\frac{\|w\|^2}{2\sigma^2}\left(\frac{2\sigma^2-\gamma^2}{\sigma^2}\right)\\
&= -\frac{\|w\|^2}{2\sigma^4}\left(2\sigma^2-\gamma^2\right),
\end{aligned}$$

we get:

$$\begin{aligned}
\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}} e^{-\frac{\langle u,w\rangle}{\sigma^2}}\mathrm{d}u\mathrm{d}w &= \left(\sqrt{2\pi\gamma^2}\right)^n\int_{\mathbb{R}^n} e^{-\frac{\|w\|^2}{2\sigma^4}\left(2\sigma^2-\gamma^2\right)}\mathrm{d}w\\
&= \left(\sqrt{2\pi\gamma^2}\right)^n\left(\sqrt{2\pi\frac{\sigma^4}{2\sigma^2-\gamma^2}}\right)^n\\
&= (2\pi)^n\left(\frac{\gamma^2\sigma^4}{2\sigma^2-\gamma^2}\right)^{n/2}.
\end{aligned}$$

□

Lemma C.2. *Consider $\sigma > 0$ and $\gamma > 0$. Denote I_n the identity matrix in $\mathbb{R}^n$. Then:*

$$\mathop{\mathbb{E}}_{w\sim\mathcal{N}(0,\sigma^2 I_n)}\mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,\sigma^2 I_n)}\left[e^{-\|u-w\|^2/2\gamma^2}\right] = \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-n/2}. \tag{40}$$

Proof. The expectation is a straightforward integral:

$$\begin{aligned}
\mathop{\mathbb{E}}_{w\sim p}\mathop{\mathbb{E}}_{u\sim p}[\mathcal{K}(u,w)] &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n}\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}}e^{-\frac{\|u\|^2}{2\sigma^2}}e^{-\frac{\|w\|^2}{2\sigma^2}}\mathrm{d}u\mathrm{d}w \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n}\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}}e^{-\frac{\|u\|^2}{2\sigma^2}}e^{\frac{\langle u,w\rangle}{\sigma^2}}e^{-\frac{\|w\|^2}{2\sigma^2}}e^{-\frac{\langle u,w\rangle}{\sigma^2}}\mathrm{d}u\mathrm{d}w \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n}\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\gamma^2}}e^{-\frac{\|u-w\|^2}{2\sigma^2}}e^{-\frac{\langle u,w\rangle}{\sigma^2}}\mathrm{d}u\mathrm{d}w \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n}\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\frac{\|u-w\|^2}{2\zeta^2}}e^{-\frac{\langle u,w\rangle}{\sigma^2}}\mathrm{d}u\mathrm{d}w & \left(\tfrac{1}{2\zeta^2}=\tfrac{1}{2\gamma^2}+\tfrac{1}{2\sigma^2}=\tfrac{\sigma^2+\gamma^2}{2\sigma^2\gamma^2}\right) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{2n}(2\pi)^n\left(\frac{\zeta^2\sigma^4}{2\sigma^2-\zeta^2}\right)^{n/2} & \text{(Lemma C.1)} \\
&= \left(\frac{\zeta^2}{2\sigma^2-\zeta^2}\right)^{n/2} \\
&= \left(\frac{2\sigma^2}{\zeta^2}-1\right)^{-n/2} \\
&= \left(\frac{2\sigma^2}{\sigma^2}+\frac{2\sigma^2}{\gamma^2}-1\right)^{-n/2} \\
&= \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-n/2}.
\end{aligned}$$

□

Lemma C.3. *Considering RWRelu, we have:*

$$\kappa = \frac{\sigma}{\sqrt{2}}\sup_{x\in\mathcal{X}}\|x\|.$$

Proof. We have:

$$\begin{aligned}
\kappa^2 &:= \sup_{x\in\mathcal{X}}\mathop{\mathbb{E}}_{w\sim p}\left[\mathcal{K}(w,w)\phi(w,x)^2\right] \\
&= \sup_{x\in\mathcal{X}}\mathop{\mathbb{E}}_{w\sim p}\left[\max\left(0,\langle w,x\rangle\right)^2\right] \\
&= \sup_{x\in\mathcal{X}}\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n\int_{\mathbb{R}^n} e^{-\|w\|^2/2\sigma^2}\max\left(0,\langle w,x\rangle\right)^2\mathrm{d}w \\
&= \sup_{x\in\mathcal{X}}\frac{1}{\sqrt{2\pi\sigma^2}}\int_{\mathbb{R}} e^{-w_n^2/2\sigma^2}\max(0,w_n\|x\|)^2\mathrm{d}w_n & \left(w_n := \tfrac{\langle w,x\rangle}{\|x\|}\text{, see proof of Lemma B.4}\right) \\
&= \sup_{x\in\mathcal{X}}\frac{1}{\sqrt{2\pi\sigma^2}}\int_0^\infty e^{-w_n^2/2\sigma^2}w_n^2\|x\|^2\mathrm{d}w_n \\
&= \sup_{x\in\mathcal{X}}\|x\|^2\frac{1}{\sqrt{2\pi\sigma^2}}\int_0^\infty e^{-w_n^2/2\sigma^2}w_n^2\mathrm{d}w_n \\
&= \frac{1}{2}\sup_{x\in\mathcal{X}}\|x\|^2\mathop{\mathbb{E}}_{w_n\sim\mathcal{N}(0,\sigma^2)}\left[w_n^2\right] \\
&= \frac{\sigma^2}{2}\sup_{x\in\mathcal{X}}\|x\|^2.
\end{aligned}$$

We have the result by taking the square root. □

Lemma C.4. *Considering RWRelu, we have:*

$$\theta^2 \leq \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{\sigma^2}{2\pi}\right). \tag{41}$$

Proof. We have:

$$\begin{aligned}\theta^2 :&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{u\sim p} \mathop{\mathbb{E}}_{w\sim p} [\mathcal{K}(u,w)\phi(u,x)\phi(w,x)] \\ &= \sup_{x\in\mathcal{X}} \left(\frac{1}{2\pi\sigma^2}\right)^n \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\|u-w\|^2/2\gamma^2} e^{-\|u\|^2/2\sigma^2} e^{-\|w\|^2/2\sigma^2} \max(0,\langle u,x\rangle)\max(0,\langle w,x\rangle)\mathrm{d}u\mathrm{d}w.\end{aligned}$$

We are free to solve these integrals using any orthonormal basis of $\mathbb{R}^n$. We choose a basis $\{v_1,\ldots,v_n\}$ such that:

$$v_n := \frac{x}{\|x\|}. \tag{42}$$

In particular, we have:

$$\begin{aligned} u_n :&= \left\langle u, \frac{x}{\|x\|}\right\rangle, \\ w_n :&= \left\langle w, \frac{x}{\|x\|}\right\rangle, \\ \langle u,x\rangle &= u_n\|x\|, \\ \langle w,x\rangle &= w_n\|x\|. \end{aligned}$$

Denoting $u_{1:n-1} := (u_1,\ldots,u_{n-1})$ and $w_{1:n-1} := (w_1,\ldots,w_{n-1})$, we can rewrite the expression:

$$e^{-\|u-w\|^2/2\gamma^2} e^{-\|u\|^2/2\sigma^2} e^{-\|w\|^2/2\sigma^2} \tag{43}$$

as:

$$e^{-\|u_{1:n-1}-w_{1:n-1}\|^2/2\gamma^2} e^{-\|u_{1:n-1}\|^2/2\sigma^2} e^{-\|w_{1:n-1}\|^2/2\sigma^2} e^{-(u_n-w_n)^2/2\gamma^2} e^{-u_n^2/2\sigma^2} e^{-w_n^2/2\sigma^2}. \tag{44}$$

This allows us to separate the integral into two parts. The first one is the integral over $(u_1,\ldots,u_{n-1})$ and $(w_1,\ldots,w_{n-1})$:

$$\left(\frac{1}{2\pi\sigma^2}\right)^{n-1} \int_{\mathbb{R}^{n-1}}\int_{\mathbb{R}^{n-1}} e^{-\|u_{1:n-1}-w_{1:n-1}\|^2/2\gamma^2} e^{-\|u_{1:n-1}\|^2/2\sigma^2} e^{-\|w_{1:n-1}\|^2/2\sigma^2} \mathrm{d}u_{1:n-1}\mathrm{d}w_{1:n-1}. \tag{45}$$

Lemma C.2 tells us that the previous expression is equal to:

$$\left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2}. \tag{46}$$

The second integral is the one over u_n and w_n:

$$\left(\frac{1}{2\pi\sigma^2}\right) \int_{\mathbb{R}}\int_{\mathbb{R}} e^{-(u_n-w_n)^2/2\gamma^2} e^{-u_n^2/2\sigma^2} e^{-w_n^2/2\sigma^2} \max(0,u_n\|x\|)\max(0,w_n\|x\|)\mathrm{d}u_n\mathrm{d}w_n. \tag{47}$$

We can simplify the integral by noticing that $\max(0,u_n) = u_n$ if $u_n \geq 0$, and 0 otherwise:

$$\left(\frac{1}{2\pi\sigma^2}\right)\|x\|^2 \int_0^\infty\int_0^\infty e^{-(u_n-w_n)^2/2\gamma^2} e^{-u_n^2/2\sigma^2} e^{-w_n^2/2\sigma^2} u_n w_n \mathrm{d}u_n\mathrm{d}w_n. \tag{48}$$

The expression for θ^2 has now become:

$$\begin{aligned}\theta^2 &= \sup_{x\in\mathcal{X}} \left(\frac{1}{2\pi\sigma^2}\right)^n \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} e^{-\|u-w\|^2/2\gamma^2} e^{-\|u\|^2/2\sigma^2} e^{-\|w\|^2/2\sigma^2} \max(0,\langle u,x\rangle)\max(0,\langle w,x\rangle)\mathrm{d}u\mathrm{d}w \\ &= \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{1}{2\pi\sigma^2}\right) \int_0^\infty\int_0^\infty e^{-(u_n-w_n)^2/2\gamma^2} e^{-u_n^2/2\sigma^2} e^{-w_n^2/2\sigma^2} u_n w_n \mathrm{d}u_n\mathrm{d}w_n. \end{aligned} \tag{49}$$

The remaining integral is difficult. We can simplify by using the fact that $e^{-(u_n-w_n)^2/2\gamma^2} \leq 1$, and obtain an upper bound on θ^2:

$$
\begin{aligned}
\theta^2 &= \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{1}{2\pi\sigma^2}\right) \int_0^\infty \int_0^\infty e^{-(u_n-w_n)^2/2\gamma^2} e^{-u_n^2/2\sigma^2} e^{-w_n^2/2\sigma^2} u_n w_n \mathrm{d}u_n \mathrm{d}w_n \\
&\leq \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{1}{2\pi\sigma^2}\right) \int_0^\infty \int_0^\infty e^{-u_n^2/2\sigma^2} e^{-w_n^2/2\sigma^2} u_n w_n \mathrm{d}u_n \mathrm{d}w_n \\
&= \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{1}{2\pi\sigma^2}\right) \left(\int_0^\infty e^{-w_n^2/2\sigma^2} w_n \mathrm{d}w_n\right)^2 \\
&= \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{1}{2\pi\sigma^2}\right) \left(\left(-\sigma^2 e^{-w_n^2/2\sigma^2}\right|_{w_n=0}^{\infty}\right)^2 \\
&= \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{1}{2\pi\sigma^2}\right) \sigma^4 \\
&= \sup_{x\in\mathcal{X}} \|x\|^2 \left(1+\frac{2\sigma^2}{\gamma^2}\right)^{-(n-1)/2} \left(\frac{\sigma^2}{2\pi}\right).
\end{aligned}
$$

□

D Theoretical constants of STAR RKHS Weightings

We showed in Section 4 how to transform an RKHS Weighting instantiation into a STAR model, while maintaining the form of the model as an RKHS Weighting (Table 2 contains the details). In particular, all algorithms and theoretical guarantees from Dubé & Marchand (2024) still apply. The guarantees involve the theoretical constants:

$$\kappa := \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\left[\|\phi(w,x)\mathcal{K}(w,\cdot)\|_{\mathcal{H}}^2\right]} = \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\left[\mathcal{K}(w,w)\phi(w,x)^2\right]} \tag{50}$$

$$\theta := \sup_{x\in\mathcal{X}} \left\|\mathop{\mathbb{E}}_{w\sim p}\left[\phi(w,x)\mathcal{K}(w,\cdot)\right]\right\|_{\mathcal{H}} = \sup_{x\in\mathcal{X}} \sqrt{\mathop{\mathbb{E}}_{w\sim p}\mathop{\mathbb{E}}_{u\sim p}\left[\mathcal{K}(u,w)\phi(u,x)\phi(w,x)\right]}, \tag{51}$$

defined for any instantiation $(\mathcal{W},\phi,\mathcal{K},p)$.

Here, we explain how to obtain those constants, if desired, for a STAR RKHS Weighting instantiation. For κ, we have:

$$
\begin{aligned}
\kappa^2 &= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{(w,I)\sim p_{\mathcal{W}|I}\times p_{[n]}} \left[\mathcal{K}((w,I),(w,I))\phi_I(w,x_I)^2\right] \\
&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{I\sim p_{[n]}} \mathop{\mathbb{E}}_{w\sim p_{\mathcal{W}|I}} \left[\mathcal{K}_W(w,w)\mathbb{1}[I=I]\phi_I(w,x_I)^2\right] \\
&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{I\sim p_{[n]}} \mathop{\mathbb{E}}_{w\sim p_{\mathcal{W}|I}} \left[\mathcal{K}_W(w,w)\phi_I(w,x_I)^2\right] \\
&\leq \mathop{\mathbb{E}}_{I\sim p_{[n]}} \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{w\sim p_{\mathcal{W}|I}} \left[\mathcal{K}_W(w,w)\phi_I(w,x_I)^2\right]
\end{aligned}
$$

The expression $\sup_{x\in\mathcal{X}} \mathbb{E}_{w\sim p_{\mathcal{W}|I}}\left[\mathcal{K}_W(w,w)\phi_I(w,x_I)^2\right]$ is in fact the constant κ^2 for the partial instantiation defined only on the feature subset I. Let's denote it κ_I^2. Then we have:

$$\kappa^2 \leq \mathop{\mathbb{E}}_{I\sim p_{[n]}} \kappa_I^2.$$

Calculating this quantity will depend on $p_{[n]}$. In the simplest cases, where for example $p_{[n]}$ is the uniform distribution on all feature subsets of size k, then the constant κ_I might simply be the same for all I. In the worst cases, we can take the supremum of κ_I over all possible subsets I instead of the expectation. As for θ:

$$\begin{aligned}
\theta^2 &= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{(w,I)\sim p_{\mathcal{W}|I}\times p_{[n]}} \mathop{\mathbb{E}}_{(u,J)\sim p_{\mathcal{W}|J}\times p_{[n]}} [\mathcal{K}((u,J),(w,I))\phi_J(u,x_J)\phi_I(w,x_I)] \\
&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{(w,I)\sim p_{\mathcal{W}|I}\times p_{[n]}} \mathop{\mathbb{E}}_{(u,J)\sim p_{\mathcal{W}|J}\times p_{[n]}} [\mathcal{K}_{\mathcal{W}}(u,w)\mathbb{1}[I=J]\phi_J(u,x_J)\phi_I(w,x_I)] \\
&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{(w,I)\sim p_{\mathcal{W}|I}\times p_{[n]}} \mathop{\mathbb{E}}_{u\sim p_{\mathcal{W}|I}} \left[p_{[n]}(I)\mathcal{K}_{\mathcal{W}}(u,w)\phi_I(u,x_I)\phi_I(w,x_I)\right] \\
&= \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{I\sim p_{[n]}} p_{[n]}(I) \mathop{\mathbb{E}}_{w\sim p_{\mathcal{W}|I}} \mathop{\mathbb{E}}_{u\sim p_{\mathcal{W}|I}} [\mathcal{K}_{\mathcal{W}}(u,w)\phi_I(u,x_I)\phi_I(w,x_I)] \\
&\leq \mathop{\mathbb{E}}_{I\sim p_{[n]}} p_{[n]}(I) \sup_{x\in\mathcal{X}} \mathop{\mathbb{E}}_{w\sim p_{\mathcal{W}|I}} \mathop{\mathbb{E}}_{u\sim p_{\mathcal{W}|I}} [\mathcal{K}_{\mathcal{W}}(u,w)\phi_I(u,x_I)\phi_I(w,x_I)].
\end{aligned}$$

Exactly as before, the expression $\sup_{x\in\mathcal{X}} \mathbb{E}_{w\sim p_{\mathcal{W}|I}} \mathbb{E}_{u\sim p_{\mathcal{W}|I}} \left[p_{[n]}(I)\mathcal{K}_{\mathcal{W}}(u,w)\phi_I(u,x_I)\phi_I(w,x_I)\right]$ is the constant θ^2 for the partial model defined on the subset I. We can write it as θ_I^2 and:

$$\theta^2 \leq \mathop{\mathbb{E}}_{I\sim p_{[n]}} p_{[n]}(I)\theta_I^2.$$

E Details of experimentation

Preprocessing of the datasets

All datasets have been scaled to have mean 0 and standard deviation 1 on all variables, including the target labels. Means and standard variations were calculated on the training data, then the transformation applied to both training and test datasets.

Numerical stability of the learning algorithm

To learn RKHS Weightings, we used Algorithm 3 of Dubé & Marchand (2024), the least squares fit of the coefficients. However, we have run into some rare numerical instability problems which led to failure of learning and even negative training R^2 scores, which is impossible in theory (at worst, the algorithm could output $0 \in \mathcal{H}$, which has an R^2 of 0 exactly). To solve this issue, we added a very small ℓ^2 regularizer to the optimization objective (Equation 53 of Dubé & Marchand (2024)), which is now:

$$\mathcal{L}_{\mathcal{S}}^{\text{reg}}(\Lambda\alpha(x)) = \frac{1}{2m}\|\Phi a - \mathbf{y}\|_2^2 + \frac{\lambda}{2}a^\top G a + \frac{\epsilon}{2}a^\top I a, \tag{52}$$

where I is the identity matrix, and ϵ is the ℓ^2 regularization parameter, which we have set to 10^{-10}. The minimizer a of the previous expression is the solution to the linear problem:

$$\left(\Phi^\top\Phi + m\lambda G + m\epsilon I\right)a = \Phi^\top \mathbf{y}. \tag{53}$$

Hyperparameter selection

As pointed out in Dubé & Marchand (2024), instantiation RWSign, and now also RWRelu, exhibits exponential behavior in the dimensionality n of the instance space. This can immediately be seen in the $c := \left(1 + \frac{\sigma^2}{\gamma^2}\right)^{-n/2}$ coefficient in Equation (18), shared by both RWSign and RWRelu. Instead of using σ and γ as hyperparameters of the model, they use σ and c, and calculate γ from σ and c. This way, the effect of dimensionality is held constant regardless of n.

We modify this methodology slightly. We instead use $c^2 := \left(1 + \frac{2\sigma^2}{\gamma^2}\right)^{-n/2}$. This is the upper bound for θ^2 calculated by Dubé & Marchand (2024) for RWSign. It can also be found in Lemma C.2, and almost as is in the upper bound for θ for RWRelu (Lemma C.4). Given σ and $c \in (0,1)$, we get:

$$\gamma^2 = \frac{2}{\sigma^2(c^{-4/n} - 1)}. \tag{54}$$

	Cross-validation parameters	Source code (clickable)
DecisionTreeRegressor	max depth $\in \{2, 5, 10, 20\}$	Scikit-learn
EBM	max bins $\in \{512, 1024, 2048\}$ learning rate $\in \{512, 1024, 2048\}$ max rounds $\in \{15000, 25000, 35000\}$ min samples leaf $\in \{1, 2, 3\}$	InterpretML
KernelRidge	kernel = rbf alpha $\in \{0.01, 0.05, 0.1, 0.5, 1, 5\}$	Scikit-learn
LinearRegression		Scikit-learn
SVR	C $\in \{0.5, 1, 5, 10, 50\}$	Scikit-learn
RWSign	max theta $\in \{0.1, 0.5, 0.9\}$ $\lambda \in \{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$	Dubé & Marchand (2024)
RWRelu	max theta $\in \{0.1, 0.5, 0.9\}$ $\lambda \in \{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$	Supplementary material
RWStumps	$\sigma \in \{0.01, 0.1, 1\}$ $\gamma \in \{0.01, 0.1, 1\}$ $\lambda \in \{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$	Dubé & Marchand (2024)

Table 7: Algorithms used in this paper and their hyperparameters.

The advantage of this method is that the model output $\Lambda\alpha(x)$ being upper bounded by $\theta\|\alpha\|_{\mathcal{H}} \leq c\|\alpha\|_{\mathcal{H}}$, this gives meaning to the parameter c. For RWSign, this allows setting the maximum value of the model. For RWRelu, the relationship is functionally the same, though the upper bound for θ is slightly more complex.

Finally, Table 7 contains the hyperparameters of all the algorithms used in this paper's experimentations.

Reproducing the results

The code for this paper can be found in the supplementary material. The repository contains a `requirements.txt` file containing the precise Python environment that was used to run the experiments. They can be installed from the command line with the command:

```
pip install -r requirements.txt
```

The experiment to produce Table 4 can be run with the command:

```
python .\major_experiments\regression.py --final
```

It will produce the result file:

`.\results\regression-final.csv`

and the table:

`.\tables\regression-final.tex`

The experiment to produce Table 6 can be run with the command:

```
python .\major_experiments\time_series.py --final
```

It will produce the result file:

`.\results\time-series-final.csv`

and the table:

```
.\tables\time-series-final.tex
```

The experiment to produce Figure 1 can be run with the command:

```
python .\major_experiments\shapley_time.py
```

It will produce the figure:

```
.\figures\shapley-time.pdf
```