Knowledge-Guided Additive Modeling for Supervised Regression

Yann Claes(✉), Vân Anh Huynh-Thu, and Pierre Geurts

University of Liège, 4000 Liège, Belgium
{y.claes,vahuynh,p.geurts}@uliege.be

Abstract. Learning processes by exploiting restricted domain knowledge is an important task across a plethora of scientific areas, with more and more hybrid methods combining data-driven and model-based approaches. However, while such hybrid methods have been tested in various scientific applications, they have been mostly tested on dynamical systems, with only limited study about the influence of each model component on global performance and parameter identification. In this work, we assess the performance of hybrid modeling against traditional machine learning methods on standard regression problems. We compare, on both synthetic and real regression problems, several approaches for training such hybrid models. We focus on hybrid methods that additively combine a parametric physical term with a machine learning term and investigate model-agnostic training procedures. We also introduce a new hybrid approach based on partial dependence functions. Experiments are carried out with different types of machine learning models, including tree-based models and artificial neural networks. Our Python implementations of the hybrid methods are available at https://github.com/yannclaes/kg-regression.

Keywords: Knowledge-guided machine learning · Physics-guided machine learning · Supervised regression · Tree-based methods · Neural networks · Partial dependence plot · Hybrid modeling

1 Introduction

For the past decades, machine learning (ML) models have been developed to tackle a variety of real-life problems, complementing/replacing model-based (MB) approaches, which mostly remain approximations that make stringent assumptions about the system under study. Traditional ML approaches are said to be *data-driven*, i.e. their prediction model is solely built from some learning

This work was supported by Service Public de Wallonie Recherche under Grant No. 2010235 - ARIAC by DIGITALWALLONIA4.AI. Computational resources have been provided by the Consortium des Equipements de Calcul Intensif (CECI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

A. Bifet et al. (Eds.): DS 2023, LNAI 14276, pp. 64–78, 2023.
https://doi.org/10.1007/978-3-031-45275-8_5

dataset, let it be (deep) neural networks or regression trees. While their design comes with great expressiveness, they are likely to be subject to over-fitting without enough training examples and to show a lack of robustness on unseen samples, with predictions that can be inconsistent w.r.t. domain knowledge [5,6,31]. To overcome this generalization issue, hybrid approaches have been introduced to incorporate *a priori* domain knowledge within statistical models, which can be leveraged in a multitude of ways (see [16,25,27] for reviews). The success of these hybrid methods have been shown empirically on a range of synthetic and real-world problems [1,7,18,31]. However, while these models have been mostly applied to dynamical systems, they have not been thoroughly studied in the context of standard regression problems. Furthermore, the majority of ML models that have been considered in these approaches are neural networks and variants of the latter, leaving aside other methods. Our contributions are the following:

- We investigate empirically the performance and benefits of hybrid methods against data-driven methods on *static* regression problems (in opposition to dynamical problems). The static context removes a layer of complexity related to the temporal correlation between observed states, which makes it easier to assess the impact and interaction between the MB and ML components. Specifically, we focus on hybrid models that combine in an additive way a parametric physical term with an ML term.
- We compare different approaches for training such hybrid additive models. We highlight specific assumptions under which these approaches are expected to work well and relate the differences in terms of prediction and parameter recovery performance. We focus on model-agnostic approaches, where the ML term can be of any type, and we compare tree-based methods against neural networks. Tree-based methods have several advantages over neural networks, which motivate their use on static regression problems: they have much less hyperparameters to tune, appear robust to the presence of irrelevant features and have been shown to outperform neural networks on tabular data [12].
- We introduce a new hybrid approach based on partial dependence functions, which makes it easier to find the right balance between the MB and ML components, makes less assumptions than other approaches, and is shown to be competitive in our experiments.

2 Problem Statement

Let us define a regression problem, with $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^d$, with $d \in \mathbb{N}_+$, drawn from a distribution $p(\mathbf{x}, y)$ such that $y = f(\mathbf{x}) + \varepsilon$ with $f : \mathbb{R}^d \mapsto \mathbb{R}$ the partially known generating function and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ the noise term. We focus on problems such that $f(\mathbf{x})$ can be decomposed as:

Assumption 1 (A1, Additivity)

$$y = f_k(\mathbf{x}_k) + f_a(\mathbf{x}) + \varepsilon,$$

where $\mathbf{x}_k$ is a subset of $K \leq d$ input variables. We assume partial knowledge of the generating function through some known algebraic function $h_k^{\theta_k}(\mathbf{x}_k) \in \mathcal{H}_k$ with tunable parameters θ_k, such that for the optimal parameters θ_k^* we have $h_k^{\theta_k^*} = f_k$. The residual term $f_a(\mathbf{x})$ is unknown and is approximated in this work through an ML component $h_a^{\theta_a} \in \mathcal{H}_a$, with parameters θ_a[1]. The final model $h \in \mathcal{H}$ is denoted $h(\mathbf{x}) = h_k^{\theta_k}(\mathbf{x}_k) + h_a^{\theta_a}(\mathbf{x})$, with the function space $\mathcal{H}$ defined as $\mathcal{H}_k + \mathcal{H}_a$. A1 is common when MB methods and ML models are combined [7,22,26,30].

Given a learning sample of N input-output pairs $LS = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, drawn from $p(\mathbf{x}, y)$, we seek to identify a function $h = h_k^{\theta_k} + h_a^{\theta_a}$, i.e. parameters θ_k and θ_a, that minimizes the following two distances:

$$d(h, y) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}\{(h(\mathbf{x}) - y)^2\}, \tag{1}$$

$$d_k(h_k^{\theta_k}, f_k) = \mathbb{E}_{\mathbf{x}_k \sim p(\mathbf{x}_k)}\{(h_k^{\theta_k}(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\}. \tag{2}$$

The first distance measures the standard generalization error of the global model h. The hope is that taking h_k into account will help learning a better global model than fitting directly a pure data-driven model on y, especially in the small sample size regime. The second distance d_k measures how well the tuned h_k approximates f_k. The main motivation for this second objective is interpretability: one expects that the algebraic form of h_k will be derived from first principles by domain experts, who will be interested in estimating the parameters of this term from data. An alternative to d_k is a loss that would compare the estimated and optimal parameters $\hat{\theta}_k$ and θ_k^* (e.g., $||\hat{\theta}_k - \theta_k^*||^2$). d_k however has the advantage not to require θ_k^* to be fully identifiable, i.e. there can exist several sets of parameters θ_k^* such that $h_k^{\theta_k^*} = f_k$. In our experiment, we will report both d_k and the relative mean absolute error on the estimated parameters.

The following approximation of (1) can be used as training objective:

$$\hat{d}(h, y; LS) = \frac{1}{N}\sum_{i=1}^{N}(h(\mathbf{x}_i) - y_i)^2. \tag{3}$$

Minimizing the distance in (2) is expected to be challenging and sometimes even ill-posed. Indeed, if h_a is too powerful, it could capture f entirely and leave little room for the estimation of f_k. Finding the right balance between h_k and h_a is thus very challenging, if not impossible, using only guidance of the learning sample LS. Unlike (1), (2) cannot be estimated from a sample of input-output pairs and hence cannot be explicitly used to guide model training. There are however several scenarios that will make the problem easier. In the following, we will discuss the optimality of the hybrid methods under two additional assumptions:

Assumption 2 (A2, Disjoint features). *Let $\mathbf{x}_a$ be a subset of features disjoint from $\mathbf{x}_k$ ($\mathbf{x}_k \cap \mathbf{x}_a = \emptyset$). There exists a function $f_a^r(\mathbf{x}_a)$ such that $f_a(\mathbf{x}) = f_a^r(\mathbf{x}_a)$ for all $\mathbf{x}$.*

[1] In the following, $h_k^{\theta_k}$ and $h_a^{\theta_a}$ will sometimes be denoted simply as h_k and h_a to lighten the notations.

Assumption 3 (A3, Independence). *Features in* $\mathbf{x}_k$ *are independent from features in* $\mathbf{x}_a$ *(*$\mathbf{x}_k \perp\!\!\!\perp \mathbf{x}_a$*).*

A2 makes the problem easier as f_k captures all the dependence of y on $\mathbf{x}_k$. In the absence of A3, it might be hard to distinguish real contributions from $\mathbf{x}_k$ to f from those due to correlations with features not in $\mathbf{x}_k$.

3 Related Work

Hybrid additive modeling methods emerged several decades ago, combining first-principles models with different ML models. Already in the 1990's, approaches in [14,17,24] complemented physics-based models with neural networks, weighting contributions of both components (e.g. through radial basis function networks), to achieve enhanced physical consistency with better generalization properties. More recently, other works applied the same principles to model dynamical systems in various domains, still massively relying on neural networks [21,22,26,30]. In a more standard regression setting, [3,32,33] combined a linear parametric term with a tree-based ML term.

Previous works have introduced regularization of the ML term to reduce parameter identification issues in the decomposition [13,15,28]. Further works on this matter introduced physically-motivated constraints in the learning objective to better control contributions of the MB/ML components [7,31]. Elements of discussion about the well-posedness of this additive decomposition have been introduced in previous works: [31] showed the existence and uniqueness of an optimal pair (h_k, h_a) when the contributions of h_a are constrained to be minimal, and [7] demonstrated the convergence of an algorithm alternating between the optimization of h_k and the optimization of h_a, without however any guarantee about convergence points.

4 Methods

We focus on model-agnostic approaches, i.e. that can be applied with any algebraic function h_k and any type of ML model h_a. For both terms, we only assume access to training functions, respectively denoted fit^{h_k}, $\text{fit}^{h_k+\gamma}$, and fit^{h_a}, that can estimate each model parameters, respectively θ_k, (θ_k, γ) and θ_a, so as to minimize the mean squared error (MSE) over LS (see below for the meaning of γ), where parametric methods rely on gradient descent. Pseudo-codes of methods in Sects. 4.1 and 4.2 and additional illustrations are given on the paper's GitHub[2].

4.1 Sequential Training of h_k and h_a

This baseline approach first fits h_k on the observed output y, then fits h_a on the resulting residuals, as done in [33]. More precisely, we first train $h_k^{\theta_k}$ on y by introducing a constant term $\gamma \in \mathbb{R}$, such that

$$(\hat{\theta}_k, \hat{\gamma}) = \text{fit}^{h_k+\gamma}(LS). \tag{4}$$

[2] https://github.com/yannclaes/kg-regression.

Our motivation for introducing the term γ will be explained below. Afterwards, we fit h_a on the output residuals: $\hat{\theta}_a = \text{fit}^{h_a}\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_i) - \hat{\gamma})\}_{i=1}^N$.

Let $\hat{\mathcal{F}}_k$ be the set of all functions $\hat{f}_k$ mapping $\mathbf{x}_k \in \mathcal{X}_k$ to some value $y \in \mathbb{R}$, i.e. $\hat{\mathcal{F}}_k = \{\hat{f}_k : \mathcal{X}_k \mapsto \mathbb{R}\}$. Under A2 and A3, it can be shown that $\hat{f}_k^* = \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} d(\hat{f}_k, y)$ is such that $\hat{f}_k^*(\mathbf{x}_k) = f_k(\mathbf{x}_k) + C$, for every $\mathbf{x}_k \in \mathcal{X}_k$, with $C = \mathbb{E}_{\mathbf{x}_a}\{f_a^r(\mathbf{x}_a)\}$ (see Appendix A). Hence, this approach is sound at least asymptotically and justifies the introduction of γ. Note however that even under A2 and A3, we have no guarantee that this approach produces the best estimator for a finite sample size, as $f_a^r(\mathbf{x}_a) + \epsilon$ acts as a pure additive noise term that needs to be averaged out during training. The approaches described in Sects. 4.2 and 4.3 try to overcome this issue by fitting $h_k^{\theta_k}$ on corrected outputs that are expected to be closer to $f_k(\mathbf{x}_k)$. Without A2 and A3, the quality of the estimation of f_k by $h_k^{\hat{\theta}_k}$, according to (2), is not guaranteed as there are regression problems satisfying A1 such that:

$$\nexists \gamma \in \mathbb{R} : \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} d(\hat{f}_k, y) = f_k + \gamma. \tag{5}$$

An example will be given in Sect. 5.2.

4.2 Alternate Training of h_k and h_a

A hybrid additive approach was proposed in [7] that alternates between updating h_k and updating h_a, using neural networks for h_a. Such alternate training was also proposed in [3,32] with a single decision tree as h_a and a linear h_k. We include this approach in our comparison, but also investigate it with random forests [2] and tree gradient boosting [9]. $\hat{\theta}_k$ is initialized by (fully) fitting $h_k^{\theta_k} + \gamma$ on y. Then, we alternate between: (1) a single epoch of gradient descent on $h_k^{\theta_k} + \gamma$ and (2) either a single epoch for h_a (in the case of neural networks, as in [7]) or a complete fit of h_a (in the case of tree-based models).

While some theoretical results are provided in [7], convergence of the alternate method towards the optimal solution is not guaranteed in general. Despite an initialization favoring h_k, it is unclear whether a too expressive h_a will not dominate h_k and finding the right balance between these two terms, e.g. by regularizing further h_a, is challenging. Under A2 and A3 however, the population version[3] of the algorithm produces an optimal solution. Indeed, h_k will be initialized as the true f_k, as shown previously, making the residuals $y - h_k$ at the first iteration, as well as h_a, independent of $\mathbf{x}_k$. h_k will thus remain unchanged (and optimal) at subsequent iterations.

[3] i.e., assuming an infinite training sample size and consistent estimators.

4.3 Partial Dependence-Based Training of h_k and h_a

We propose a novel approach relying on partial dependence (PD) functions [8] to produce a proxy dataset depending only on $\mathbf{x}_k$ to fit h_k. PD measures how a given subset of features impact the prediction of a model, on average. Let $\mathbf{x}_k$ be the subset of interest and $\mathbf{x}_{-k}$ its complement, with $\mathbf{x}_k \cup \mathbf{x}_{-k} = \mathbf{x}$, then the PD of a function $f(\mathbf{x})$ on $\mathbf{x}_k$ is:

$$PD(f, \mathbf{x}_k) = \mathbb{E}_{\mathbf{x}_{-k}}\left[f(\mathbf{x}_k, \mathbf{x}_{-k})\right] = \int f(\mathbf{x}_k, \mathbf{x}_{-k}) p(\mathbf{x}_{-k}) d\mathbf{x}_{-k}, \tag{6}$$

where $p(\mathbf{x}_{-k})$ is the marginal distribution of $\mathbf{x}_{-k}$. Under A1 and A2, the PD of $f(\mathbf{x}) = f_k(\mathbf{x}_k) + f_a^r(\mathbf{x}_a)$ is [8]:

$$PD(f, \mathbf{x}_k) = f_k(\mathbf{x}_k) + C, \text{ with } C = E_{\mathbf{x}_a}\{f_a^r(\mathbf{x}_a)\}. \tag{7}$$

The idea of our method is to first fit any sufficiently expressive ML model $h_a(\mathbf{x})$ on LS and to compute its PD w.r.t. $\mathbf{x}_k$ to obtain a first approximation of $f_k(\mathbf{x}_k)$ (up to a constant). Although computing the actual PD of a function using (6) requires in principle access to the input distribution, an approximation can be estimated from LS as follows:

$$\widehat{PD}(h_a, \mathbf{x}_k; LS) = \frac{1}{N}\sum_{i=1}^{N} h_a(\mathbf{x}_k, \mathbf{x}_{i,-k}), \tag{8}$$

where $\mathbf{x}_{i,-k}$ denotes the values of $\mathbf{x}_{-k}$ in the i-th sample of LS. A new dataset of pairs $(\mathbf{x}_k, \widehat{PD}(h_a, \mathbf{x}_k; LS))$ can then be built to fit h_k. In our experiments, we consider only the $\mathbf{x}_k$ values observed in the learning sample but $\widehat{PD}(h_a, \mathbf{x}_k; LS)$ could also be estimated at other points $\mathbf{x}_k$ to artificially increase the size of the proxy dataset.

In practice, optimizing θ_k only once on the PD of h_a could leave residual dependence of $\mathbf{x}_k$ on the resulting $y - h_k^{\hat{\theta}_k}(\mathbf{x}_k) - \hat{\gamma}$. We thus repeat the sequence of fitting h_a on the latter residuals, then fitting h_k on the obtained $\widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_k; LS) + h_k^{\hat{\theta}_k}(\mathbf{x}_k) + \hat{\gamma}$, with $\hat{\theta}_k$ and $\hat{\theta}_a$ the current optimized parameter vectors (see Algorithm 1).

The main advantage of this approach over the alternate one is to avoid domination of h_a over h_k. Unlike the two previous approaches, this one is also sound even if A3 is not satisfied as it is not a requirement for (7) to hold. One drawback is that it requires h_a to capture well the dependence of f on $\mathbf{x}_k$ so that its PD is a good approximation of f_k. The hope is that even if it is not the case at the first iteration, fitting h_k, that contains the right inductive bias, will make the estimates better and better over the iterations.

5 Experiments

We compare the different methods on several regression datasets, both simulated and real. Performance is measured through estimates of (1) and (2) (the latter

Algorithm 1. Partial Dependence Optimization

Input: $LS = (\mathbf{x}_i, y_i)_{i=1}^N$
$\hat{\theta}_a \leftarrow \text{fit}^{h_a}(LS)$
$(\hat{\theta}_k, \hat{\gamma}) \leftarrow \text{fit}^{h_k+\gamma}(\{(\mathbf{x}_{k,i}, \widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_{k,i}; LS))\}_{i=1}^N)$
for $n = 1$ **to** $N_{repeats}$ **do**
  $\hat{\theta}_a \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_{k,i}) - \hat{\gamma})\}_{i=1}^N)$
  $(\hat{\theta}_k, \hat{\gamma}) \leftarrow \text{fit}^{h_k+\gamma}(\{(\mathbf{x}_{k,i}, h_k^{\hat{\theta}_k}(\mathbf{x}_{k,i}) + \hat{\gamma} + \widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_{k,i}; LS))\}_{i=1}^N)$
end for
$\hat{\theta}_a \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_{k,i}) - \hat{\gamma})\}_{i=1}^N)$

only on simulated problems) on a test set TS, respectively denoted $\hat{d}(h, y; TS)$ and $\hat{d}_k(h_k^{\theta_k}, f_k; TS)$. In some cases, we also report $\text{rMAE}(\theta_k^*, \theta_k)$, the relative mean absolute error between θ_k^* and θ_k (lower is better for all measures). For the hybrid approaches, we use as h_a either a multilayer perceptron (MLP), gradient boosting with decision trees (GB) or random forests (RF). We compare these hybrid models to a standard data-driven model that uses only h_a. We also compare fitting h_a with and without input filtering, i.e. respectively removing or keeping $\mathbf{x}_k$ from its inputs, to verify convergence claims about h_k in Sect. 4.2. Architectures (e.g. for MLP, the number of layers and neurons) are kept fixed across training methods to allow a fair comparison between them, and are given in Appendix B. We use early stopping of gradient descent training by monitoring the loss on a validation set (except for pure tree-based models, which are trained in the standard way, hence not using gradient descent).

5.1 Friedman Problem (A2 and A3 Satisfied)

We consider the following synthetic regression problem:

$$y = \theta_0 \sin(\theta_1 x_0 x_1) + \theta_2 (x_2 - \theta_3)^2 + \theta_4 x_3 + \theta_5 x_4 + \varepsilon,$$

where $x_j \sim \mathcal{U}(0,1), j = 0, \ldots 9$, and $\varepsilon \sim \mathcal{N}(0,1)$ [10]. We generate 10 different datasets using 10 different sets of values for $\theta_0, \ldots, \theta_5$, each with 300, 300 and 600 samples for respectively the training, validation and test sets. For the hybrid approaches, we use the first term as prior knowledge, i.e. $f_k(\mathbf{x}_k) = \theta_0 \sin(\theta_1 x_0 x_1)$.

We see in Table 1 that all hybrid training schemes outperform their data-driven counterpart. They come very close to the ideal $f_k \rightarrow h_a$ method, and sometimes even slightly better, probably due to chance. Sequential fitting of h_k and h_a performs as well as the alternate or PD-based approaches, as A2 and A3 are satisfied for this problem (see Sect. 4.1). Filtering generally improves the performance of hybrid schemes as A2 is verified. PD-based optimization yields good approximations of f_k (as shown by a low $\hat{d}_k$). The alternate approach follows closely whereas the sequential one ends up last, which can be expected as fitting h_k only on y induces a higher noise level centered around $\mathbb{E}_{\mathbf{x}_a}\{f_a(\mathbf{x}_a)\}$, while the other approaches benefit from reduced perturbations through h_a estimation,

Table 1. Results on the Friedman problem. We report the mean and standard deviation of $\hat{d}$ and $\hat{d}_k$ over the 10 test sets (TS). "$f_k \rightarrow h_a$" fits h_a on $y - f_k(\mathbf{x}_k)$. "Unfiltered" indicates that all the features are used as inputs of h_a, while "Filtered" indicates that the features $\mathbf{x}_k$ are removed from the inputs of h_a.

	Method	$\hat{d}(h, y; TS)$		$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$	
		Unfiltered	Filtered	Unfiltered	Filtered
MLP	$f_k \rightarrow h_a$	1.58 ± 0.33	1.23 ± 0.10	–	
	Sequential	1.54 ± 0.31	1.43 ± 0.13	0.18 ± 0.16	
	Alternate	1.43 ± 0.09	1.32 ± 0.09	0.10 ± 0.09	0.02 ± 0.02
	PD-based	1.54 ± 0.12	1.38 ± 0.09	0.06 ± 0.07	
	h_a only	2.62 ± 0.75		–	
GB	$f_k \rightarrow h_a$	1.73 ± 0.09	1.75 ± 0.12	–	
	Sequential	1.74 ± 0.11	1.81 ± 0.14	0.18 ± 0.16	
	Alternate	1.79 ± 0.11	1.78 ± 0.15	0.91 ± 1.45	0.06 ± 0.06
	PD-based	1.77 ± 0.13	1.78 ± 0.12	0.03 ± 0.02	
	h_a only	3.43 ± 0.94		–	
RF	$f_k \rightarrow h_a$	2.03 ± 0.18	1.96 ± 0.17	–	
	Sequential	2.11 ± 0.23	2.05 ± 0.24	0.18 ± 0.16	
	Alternate	2.03 ± 0.19	1.98 ± 0.17	0.04 ± 0.03	0.04 ± 0.04
	PD-based	2.16 ± 0.27	2.09 ± 0.26	0.16 ± 0.15	
	h_a only	5.58 ± 1.91		–	

as explained in Sect. 4.1. Filtering vastly decreases $\hat{d}_k$ for alternate approaches, supporting claims introduced in Sect. 4.2, while this measure remains unimpaired for sequential and PD-based training by construction.

5.2 Correlated Input Features (A3 Not Satisfied)

Correlated Linear Model. Let $y = \beta_0 x_0 + \beta_1 x_1 + \varepsilon$, with $\beta_0 = -0.5, \beta_1 = 1, \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, and $\varepsilon \sim \mathcal{N}(0.5^2, 1)$. We generate 50, 50 and 600 samples respectively for the training, validation and test sets. We use as known term $f_k(\mathbf{x}_k) = \beta_0 x_0$. Regressing y on x_0 yields the least-squares solution [11]:

$$\mathbb{E}\left[\hat{\beta}_0\right] = \beta_0 + \frac{\text{cov}(x_0, x_1)}{\text{var}(x_0)}\beta_1. \tag{9}$$

We set $\text{cov}(x_0, x_1) = 2.25$ and $\text{var}(x_0) = 2$ so that (9) reverses the sign of β_0 and (5) is satisfied. The sequential approach should hence yield parameter estimates of β_0 close to (9) while we expect the others to correct for this bias.

From Table 2, we observe that, contrary to the PD-based approach, the sequential and alternate methods return very bad estimations of β_0, as A3 is no longer verified. Filtering corrects the bias for the alternate approach but

Table 2. Results for the correlated linear problem. We report $\hat{d}$ and rMAE$(\beta_0^*, \hat{\beta}_0)$, over 10 different datasets.

	Method	$\hat{d}(h, y; TS)$		rMAE$(\beta_0^*, \hat{\beta}_0)$	
		Unfiltered	Filtered	Unfiltered	Filtered
MLP	Sequential	0.30 ± 0.03	0.74 ± 0.09	224.14 ± 13.48	
	Alternate	0.30 ± 0.02	0.31 ± 0.04	186.65 ± 21.31	15.53 ± 13.57
	PD-based	0.30 ± 0.03	0.29 ± 0.02	26.47 ± 17.32	
GB	Sequential	0.59 ± 0.06	1.38 ± 0.11	224.14 ± 13.48	
	Alternate	0.57 ± 0.06	0.60 ± 0.09	148.75 ± 67.35	24.58 ± 12.20
	PD-based	0.56 ± 0.05	0.64 ± 0.13	36.05 ± 17.50	
RF	Sequential	0.53 ± 0.05	0.90 ± 0.07	224.14 ± 13.48	
	Alternate	0.43 ± 0.04	0.42 ± 0.04	111.04 ± 52.78	45.38 ± 22.39
	PD-based	0.41 ± 0.03	0.43 ± 0.04	57.47 ± 15.55	

degrades the MSE performance for the sequential method as it removes the ability to compensate for the h_k misfit.

Correlated Friedman Problem. The structure is identical to the one in Sect. 5.1 but with correlated inputs drawn from a multivariate normal distribution where $\mu_i = 0.5$ and $\text{var}(x_i) = 0.75, \forall i$, and $\text{cov}(x_i, x_j) = \pm 0.3, \forall i \neq j$ (the covariance sign being chosen randomly). Sizes of the training, validation and test sets are identical to those of Sect. 5.1. Inputs are then scaled to be roughly in $[-1, 1]$. Here again, we use $f_k(\mathbf{x}_k) = \theta_0 \sin(\theta_1 x_0 x_1)$.

As in Sect. 5.1, Table 3 shows that hybrid models outperform their data-driven equivalents. PD-based methods usually yield more robust h_k estimations in the general unfiltered case, but struggle to line up with the alternate scheme in terms of predictive performance, except for GB-related models. For RF, this can be explained by a worse h_k estimation while for MLP we assume that it is due to h_a overfitting: in the alternate approach, it is optimized one epoch at a time, interleaved with one step on h_k, whereas that of PD-based methods is fully optimized (with identical complexities). Sequential and alternate approaches undergo stronger h_k misparameterization without filtering since A3 is not met, but the latter mitigates this w.r.t. the former, as was already observed in Sect. 5.1. Input filtering degrades predictive performance for the sequential methods as they cannot counterbalance a poor h_k.

5.3 Overlapping Additive Structure (A2 and A3 Not Satisfied)

Let $y = \beta x_0^2 + \sin(\gamma x_0) + \delta x_1 + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 0.5^2)$, $\beta = 0.2, \gamma = 1.5, \delta = 1$ and $\mathbf{x}$ sampled as in the correlated linear problem. We generate 50, 50 and 600 samples respectively for the training, validation and test sets. We define $f_k(\mathbf{x}_k) = \beta x_0^2$ and $f_a(\mathbf{x}) = \sin(\gamma x_0) + \delta x_1 + \varepsilon$. Hence, A2 and A3 do not hold. Even with

Table 3. Results for the correlated Friedman problem.

	Method	$\hat{d}(h, y; TS)$		$\hat{d}_k(h_k^{\theta_k}, f_k; TS)$	
		Unfiltered	Filtered	Unfiltered	Filtered
MLP	$f_k \rightarrow h_a$	1.64 ± 0.23	1.51 ± 0.17	–	
	Sequential	2.07 ± 0.40	2.68 ± 1.38	1.35 ± 1.42	
	Alternate	1.95 ± 0.33	1.62 ± 0.24	0.49 ± 0.44	0.14 ± 0.19
	PD-based	2.24 ± 0.31	1.78 ± 0.30	0.17 ± 0.23	
	h_a only	2.77 ± 0.73		–	
GB	$f_k \rightarrow h_a$	2.58 ± 0.45	2.53 ± 0.44	–	
	Sequential	2.90 ± 0.39	3.91 ± 1.49	1.35 ± 1.42	
	Alternating	2.67 ± 0.38	2.62 ± 0.43	0.51 ± 0.53	0.22 ± 0.25
	PD-based	2.54 ± 0.35	2.47 ± 0.36	0.03 ± 0.02	
	h_a only	4.49 ± 0.66		–	
RF	$f_k \rightarrow h_a$	3.02 ± 0.45	2.93 ± 0.45	–	
	Sequential	3.78 ± 0.78	4.04 ± 1.30	1.35 ± 1.42	
	Alternating	3.06 ± 0.39	2.99 ± 0.38	0.14 ± 0.16	0.15 ± 0.18
	PD-based	3.24 ± 0.38	3.16 ± 0.37	0.27 ± 0.20	
	h_a only	6.70 ± 1.47		–	

$\hat{\beta} = \beta^*$, h_a still needs to compensate for $\sin(\gamma x_0)$. Filtering is thus expected to degrade performance for all hybrid approaches as $h_a(x_1)$ will never compensate this gap, which is observed in Table 4. Results for RF are not shown for the sake of space, but are similar to GB.

Table 4. Results for the overlapping problem.

Method	$\hat{d}(h, y; TS)$		$\hat{d}(h, y; TS)$	
	Unfiltered	Filtered	Unfiltered	Filtered
	MLP		GB	
$f_k \rightarrow h_a$	0.35 ± 0.02	0.54 ± 0.04	0.51 ± 0.04	1.00 ± 0.12
Sequential	0.35 ± 0.01	0.59 ± 0.05	0.55 ± 0.07	1.07 ± 0.11
Alternate	0.35 ± 0.02	0.56 ± 0.05	0.54 ± 0.09	1.01 ± 0.11
PD-based	0.34 ± 0.02	0.56 ± 0.05	0.53 ± 0.05	0.99 ± 0.12
h_a only	0.37 ± 0.02		0.55 ± 0.07	

5.4 Real Regression Problems

We now apply all methods on two real-world static datasets. As algebraic term h_k, we chose to use a linear prior on x_k, where x_k is the feature with the highest

importance score in a RF model trained on the full dataset (assumed to be inaccessible at training time). As there is no guarantee that any of our assumptions are met (and in particular A1), we do not measure the distance d_k in (2) as it is deemed irrelevant.

We consider two settings for each dataset, inspired by [33]. In the first setting (INT), the training and test sets are sampled from the same distribution $p(\mathbf{x}, y)$ whereas the second one (EXT) evaluates extrapolation performance for samples with unseen target values. If the linear prior is a reasonable assumption or, at the very least, the target increases or decreases monotonically with x_k, then we can expect hybrid methods to yield better results in the latter setting. In the INT setting for each dataset, we randomly select 100 samples for the learning set, 100 samples for the validation set and keep the rest as test set. For the EXT setting, we select the samples (one fourth of the dataset) with the lowest output values as test set. From the remaining samples, we randomly select 100 samples for the learning set and 100 samples for the validation set. For both INT and EXT settings, performance metrics are averaged over 10 different splits. We standardize both input and output variables.

The features for both datasets are described in Table 5. The *Combined Cycle Power Plant* (CCPP) dataset [23] collects 9,568 measurements of net hourly electrical energy output for a combined cycle power plant, along with four hourly average input variables. The *Concrete Compressive Strength (CCS) dataset* [29] is composed of 1,030 samples relating amounts of concrete components with the resulting compressive strength. As done in [33], we introduce a new feature corresponding to the cement-to-water ratio.

Table 5. Variables used for the real-world datasets. For each dataset, the variable indicated in bold type is the one used in the linear prior (x_k).

Dataset	Name	Description
CCPP	**T**	**Ambient temperature [°C]**
	AP	Ambient pressure [mbar]
	RH	Relative humidity [-]
	V	Exhaust vacuum [cmHg]
CCS	Cement	Amount of cement in the mixture [kg/m^3]
	Blast Furnace Slag	Amount of blast furnace slag in the mixture [kg/m^3]
	Fly Ash	Amount of fly ash in the mixture [kg/m^3]
	Water	Amount of water in the mixture [kg/m^3]
	Superplasticizer	Amount of superplasticizer in the mixture [kg/m^3]
	Coarse Aggregate	Amount of coarse aggregate in the mixture [kg/m^3]
	Fine Aggregate	Amount of fine aggregate in the mixture [kg/m^3]
	Age	Day (1–365)
	Cement/Water	**Cement to water ratio [-]**

Table 6. Results for the real-world datasets. We report the mean and standard deviation of $\hat{d}$ over the test sets.

	Method	CCPP		CCS	
		INT	EXT	INT	EXT
MLP	Sequential	0.07 ± 0.01	0.23 ± 0.13	0.25 ± 0.04	0.74 ± 0.24
	Alternating	0.07 ± 0.01	0.24 ± 0.10	0.24 ± 0.03	0.89 ± 0.36
	PD-based	0.07 ± 0.01	0.07 ± 0.01	0.25 ± 0.03	0.38 ± 0.04
	h_a only	0.07 ± 0.01	0.36 ± 0.21	0.25 ± 0.05	0.83 ± 0.30
GB	Sequential	0.08 ± 0.01	0.35 ± 0.07	0.21 ± 0.02	0.91 ± 0.24
	Alternating	0.08 ± 0.01	0.41 ± 0.17	0.20 ± 0.03	0.85 ± 0.28
	PD-based	0.08 ± 0.02	0.10 ± 0.01	0.22 ± 0.02	0.31 ± 0.06
	h_a only	0.10 ± 0.01	0.95 ± 0.18	0.25 ± 0.02	1.54 ± 0.28
RF	Sequential	0.07 ± 0.01	0.28 ± 0.07	0.20 ± 0.02	1.06 ± 0.31
	Alternating	0.07 ± 0.01	0.32 ± 0.12	0.20 ± 0.02	1.00 ± 0.32
	PD-based	0.07 ± 0.01	0.08 ± 0.01	0.20 ± 0.02	0.30 ± 0.08
	h_a only	0.08 ± 0.01	0.99 ± 0.19	0.25 ± 0.02	1.89 ± 0.39

From Table 6, it seems that introducing a linear prior does not yield any benefit in the interpolation setting, as all models perform equally well, which suggests that either the prior is not adequate or that A1 is not verified. In the extrapolation scenario, we can however observe that the linear prior allows to mitigate the impact of moving out of the distribution compared to data-driven models. Indeed, compared to the INT setting, the performance of all purely data-driven methods degrades in the EXT scenario, especially for GB and RF as their output predictions are bounded by the minimum target value observed in the training set. PD-based hybrid methods consistently outperform other hybrid approaches and are only slightly impacted, while sequential and alternating methods attain similar results.

6 Conclusion

We study several hybrid methods on supervised regression problems modeled in an additive way, using neural network models and tree-based approaches. We empirically show that trends observed for neural networks also apply for the non-parametric tree-based approaches, in terms of predictive performance as well as in the estimation of the algebraic known function. We introduce claims related to the convergence of these hybrid approaches, under mild assumptions, and verify their soundness on illustrative experiments. We present a new hybrid approach leveraging partial dependence and show its competitiveness against sequential and alternate optimization schemes on both synthetic and real-world problems. We highlight its benefits in estimating the parametric prior and show that it

alleviates both the risk of the ML term to dominate the known term and the need for assuming independent input features sets.

As a more general conclusion, hybrid methods are shown in our experiments to improve predictive performance with respect to ML-only models, although not always very significantly. The main benefit of the alternate and PD-based methods over the simple sequential approach is that they provide better estimators of the physical term, especially when filtering can be applied. Why this advantage does not always translate into better overall predictive performance remains to be analyzed as future work. We also plan to investigate further the theoretical properties of the PD-based approach and extend it to dynamical problems.

A Optimal Model Under A2 and A3

Let us recall the regression problem, where $y \in \mathbb{R}$ can be decomposed into the addition of two independent terms:

$$y = f_k(\mathbf{x}_k) + f_a^r(\mathbf{x}_a) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \quad \mathbf{x}_k \cup \mathbf{x}_a = \mathbf{x}, \mathbf{x}_k \cap \mathbf{x}_a = \emptyset, \mathbf{x}_k \perp\!\!\!\perp \mathbf{x}_a.$$

For clarity, let us denote by $\mathbb{E}_{\mathbf{x}}$ the subsequent expectations over the input space $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\{\cdot\}$. We have:

$$\begin{aligned}\hat{f}_k^* &= \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} d(\hat{f}_k, y) = \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x},\varepsilon}\left\{\left(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k) - f_a^r(\mathbf{x}_a) - \varepsilon\right)^2\right\} \\ &= \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x}_k}\left\{(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\right\} + \mathbb{E}_{\mathbf{x}_a,\varepsilon}\left\{(f_a^r(\mathbf{x}_a) + \varepsilon)^2\right\} \\ &\quad - \mathbb{E}_{\mathbf{x},\varepsilon}\left\{2(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))(f_a^r(\mathbf{x}_a) + \varepsilon)\right\}.\end{aligned}$$

The second term is independent w.r.t. $\hat{f}_k$ and thus has no impact on the minimization. Moreover, since $\mathbf{x}_k \perp\!\!\!\perp \mathbf{x}_a$, the last term writes as the product of two expectations, one of which is constant w.r.t. $\hat{f}_k$. We thus have:

$$\hat{f}_k^* = \arg\min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x}_k}\left\{(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\right\} - 2\,C\mathbb{E}_{\mathbf{x}_k}\left\{(\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))\right\}, \quad (10)$$

with $C = \mathbb{E}_{\mathbf{x}_a,\varepsilon}\{(f_a^r(\mathbf{x}_a) + \varepsilon)\} = \mathbb{E}_{\mathbf{x}_a}\{f_a^r(\mathbf{x}_a)\}$. Cancelling the derivative of (10) w.r.t. $\hat{f}_k$, we obtain the optimal model $\hat{f}_k^*(\mathbf{x}_k) = f_k(\mathbf{x}_k) + C$, for every $\mathbf{x}_k \in \mathcal{X}_k$.

B Model Architectures

Model hyperparameters are reported in Table 7. We used `PyTorch` [19] for MLP, `scikit-learn` [20] for RF and `xgboost` [4] for GB. Unspecified parameters keep their default values. Learning rates for training h_k and MLP are set to 0.005. H is the number of hidden layers in MLP and W the number of neurons per hidden layer. T is the number of trees in GB and RF, d the maximum tree depth and mss the minimum number of samples required to split an internal tree node.

Table 7. Model hyperparameters, for each experiment.

Problem	MLP	GB	RF
(Correlated) Friedman	$H = 2, W = 15$	$T = 700, d = 2$	$T = 500, mss = 5$
Linear & Overlap	$H = 2, W = 10$	$T = 400, d = 2$	$T = 500, mss = 5$
Real-world problems	$H = 2, W = 30$	$T = 300, d = 2$	$T = 200, mss = 5$

References

1. Ayed, I., de Bézenac, E., Pajot, A., Brajard, J., Gallinari, P.: Learning dynamical systems from partial observations. In: Machine Learning and the Physical Sciences: Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS) (2019)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
3. Chen, J., Yu, K., Hsing, A., Therneau, T.M.: A partially linear tree-based regression model for assessing complex joint gene-gene and gene-environment effects. Genet. Epidemiol. Off. Publ. Int. Genet. Epidemiol. Soc. **31**(3), 238–251 (2007)
4. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 785–794. ACM, New York (2016)
5. Daw, A., Karpatne, A., Watkins, W., Read, J., Kumar, V.: Physics-guided neural networks (PGNN): an application in lake temperature modeling. arXiv preprint arXiv:1710.11431 (2017)
6. De Bézenac, E., Pajot, A., Gallinari, P.: Deep learning for physical processes: incorporating prior scientific knowledge. J. Stat. Mech. Theory Exp. **2019**(12), 124009 (2019)
7. Donà, J., Déchelle, M., Levy, M., Gallinari, P.: Constrained physical-statistics models for dynamical system identification and prediction. In: ICLR 2022-The Tenth International Conference on Learning Representations (2022)
8. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001)
9. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
10. Friedman, J.H., Grosse, E., Stuetzle, W.: Multidimensional additive spline approximation. SIAM J. Sci. Stat. Comput. **4**(2), 291–301 (1983)
11. Greene, W.H.: Econometric Analysis. Pearson Education (2003)
12. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2022)
13. Hu, G., Mao, Z., He, D., Yang, F.: Hybrid modeling for the prediction of leaching rate in leaching process based on negative correlation learning bagging ensemble algorithm. Comput. Chem. Eng. **35**(12), 2611–2617 (2011)
14. Johansen, T.A., Foss, B.A.: Representing and learning unmodeled dynamics with neural network memories. In: 1992 American Control Conference, pp. 3037–3043. IEEE (1992)
15. Kahrs, O., Marquardt, W.: Incremental identification of hybrid process models. Comput. Chem. Eng. **32**(4–5), 694–705 (2008)

16. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nat. Rev. Phys. **3**(6), 422–440 (2021)
17. Kramer, M.A., Thompson, M.L., Bhagat, P.M.: Embedding theoretical models in neural networks. In: 1992 American Control Conference, pp. 475–479. IEEE (1992)
18. Mehta, V., et al.: Neural dynamical systems: balancing structure and flexibility in physical prediction. In: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 3735–3742. IEEE (2021)
19. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019)
20. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
21. Qian, Z., Zame, W., Fleuren, L., Elbers, P., van der Schaar, M.: Integrating expert ODEs into neural ODEs: pharmacology and disease progression. In: Advances in Neural Information Processing Systems, vol. 34, pp. 11364–11383 (2021)
22. Takeishi, N., Kalousis, A.: Physics-integrated variational autoencoders for robust and interpretable generative modeling. In: Advances in Neural Information Processing Systems, vol. 34, pp. 14809–14821 (2021)
23. Tfekci, P., Kaya, H.: Combined Cycle Power Plant. UCI Machine Learning Repository (2014). https://doi.org/10.24432/C5002N
24. Thompson, M.L., Kramer, M.A.: Modeling chemical processes using prior knowledge and neural networks. AIChE J. **40**(8), 1328–1340 (1994)
25. Von Rueden, L., et al.: Informed machine learning-a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Trans. Knowl. Data Eng. **35**(1), 614–633 (2021)
26. Wehenkel, A., Behrmann, J., Hsu, H., Sapiro, G., Louppe, G., Jacobsen, J.H.: Improving generalization with physical equations. In: Machine Learning and the Physical Sciences: Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS) (2022)
27. Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V.: Integrating scientific knowledge with machine learning for engineering and environmental systems. ACM Comput. Surv. **55**(4), 1–37 (2022)
28. Wouwer, A.V., Renotte, C., Bogaerts, P.: Biological reaction modeling using radial basis function networks. Comput. Chem. Eng. **28**(11), 2157–2164 (2004)
29. Yeh, I.C.: Concrete Compressive Strength. UCI Machine Learning Repository (2007). https://doi.org/10.24432/C5PK67
30. Yin, Y., Ayed, I., de Bézenac, E., Baskiotis, N., Gallinari, P.: LEADS: learning dynamical systems that generalize across environments. In: Advances in Neural Information Processing Systems, vol. 34, pp. 7561–7573 (2021)
31. Yin, Y., et al.: Augmenting physical models with deep networks for complex dynamics forecasting. J. Stat. Mech. Theory Exp. **2021**(12), 124012 (2021)
32. Yu, K., et al.: A partially linear tree-based regression model for multivariate outcomes. Biometrics **66**(1), 89–96 (2010)
33. Zhang, H., Nettleton, D., Zhu, Z.: Regression-enhanced random forests. arXiv preprint arXiv:1904.10416 (2019)