

KNOWLEDGE-GUIDED MACHINE LEARNING FOR SUPERVISED REGRESSION

Yann CLAES

advised by

Pierre GEURTS, Vân Anh Huynh-Thu

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

to the

Department of Electrical Engineering and Computer Science
University of Liège

August 31, 2026

On fait la science avec des faits comme une maison avec des pierres; mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison.

— Henri Poincaré (1902)

JURY

Hendrik BLOCHEEL
Professor at KU Leuven

Benoît FRENAY
Professor at the University of Namur

Pierre GEURTS (Advisor)
Professor at the University of Liège

Vân Anh HUYNH-THU (Co-advisor)
Professor at the University of Liège

Gilles LOUPPE
Professor at the University of Liège

Arnout VAN MESSEM
Professor at the University of Liège

Louis WEHENKEL (President)
Professor at the University of Liège

ABSTRACT

For centuries, humanity has collected observations about life in general, through the study of social interactions, physical bodies and their evolution, as well as many other more or less complex natural processes and their related growth and decay cycles. Altogether, these observations constitute *knowledge*, from which mankind crafted continuously evolving models as fixed representations of their current understanding of the world. With the advent of machine learning, this understanding can be broadened through its ability of discovering new models from quantities of data, a process referred to as *data-driven* modeling, in opposition to traditional knowledge creation. Although adequate if given sufficient data about the problem at hand, this automatic modeling framework generally struggles or even fails to generalize properly when the amount of collected information drastically reduces. In such scenarios, the integration of prior knowledge in the learning process of machine learning models likely improves both data efficiency and robustness; this hybrid paradigm is called *knowledge-guided* machine learning. This thesis investigates the introduction of various forms of prior knowledge into the training process of several machine learning algorithms. In particular, our work explores novel ways of training combinations of partial physical models with machine learning models through a model-agnostic procedure, which remains competitive both in terms of predictive performance as in the estimation of the physical prior. Moreover, we also investigate how prior knowledge about model explanations, in the form of partial dependence, can help to obtain models that respond better to distribution shifts and whose decision process faithfully aligns with domain knowledge. Finally, we consider the use of partial auxiliary data to yield models that are more robust under data scarcity conditions.

CONTENTS

I	Prologue	1
1	Introduction	2
1.1	Research Question	4
1.2	Outline	5
1.3	Publications	5
II	Matter	10
2	Related Work	11
2.1	Knowledge-guided machine learning	12
2.1.1	Physics-informed training loss functions	12
2.1.2	Prior knowledge as hard constraints	12
2.1.3	Physics-informed machine learning architectures	15
2.2	Explanation-guided learning	17
2.2.1	Local explanation alignment	17
2.2.2	Global explanation alignment	18
2.3	On the concept of knowledge	18
3	Hybrid Additive Modeling for Static Supervised Regression	24
3.1	Introduction	26
3.2	Related work	26
3.3	Methodology	27
3.3.1	Problem statement	27
3.3.2	Training methods	28
3.4	Experiments	31
3.5	Conclusion	38
3.A	Model architectures	41
4	Hybrid Additive Modeling for Dynamical Systems Forecasting	42
4.1	Methodology	44
4.1.1	Problem statement	44
4.1.2	Training methods	44
4.2	Experiments	47
4.2.1	<i>Lotka-Volterra</i> system	47
4.2.2	<i>Damped Pendulum</i> system	48
4.2.3	<i>Reaction-Diffusion</i> system	48
4.2.4	Impact of regularizer and model initialization	49
4.3	Conclusion	51
4.A	Model architectures	53
4.B	Dynamical experiments details	55

4.B.1	<i>Lotka-Volterra system</i>	55
4.B.2	<i>Damped Pendulum system</i>	55
4.B.3	<i>Reaction-Diffusion system</i>	56
5	Steering Neural Network Training through Interpretable Constraints Based on Partial Dependence	57
5.1	Introduction	59
5.2	Related work	59
5.3	Methodology	60
5.3.1	Partial dependence	60
5.3.2	Problem statement	61
5.3.3	Constraint configurations	63
5.4	Experiments	63
5.4.1	Friedman problem	64
5.4.2	Product problem	70
5.4.3	Concrete compressive strength prediction	73
5.4.4	PHALK dataset	74
5.4.5	Comparison to physics-informed models	76
5.4.6	Dynamical systems forecasting	82
5.4.7	Constraining with mis-specified knowledge	84
5.5	Conclusion	88
5.A	List of variables	93
5.A.1	Concrete compressive strength dataset	93
5.A.2	PHALK dataset	93
6	Learning with Small Data by Transferring Marginal Effects from Large Data	94
6.1	Introduction	96
6.2	Methodology	97
6.2.1	Problem statement	97
6.2.2	Marginal plots	97
6.2.3	Training approach	98
6.2.4	Asymptotic analysis of the loss function	99
6.3	Related work	100
6.4	Experiments	100
6.4.1	Product problem	101
6.4.2	Real-world problems	101
6.4.3	Ablation study	104
6.5	Conclusion	105
III	Epilogue	110
7	Discussion	111
7.1	Confining hybrid additive models to static supervised regression problems	111
7.2	Straying from the beaten tracks of hybrid additive modeling	113
7.3	Softening the additivity bias through explanation-guided learning	115
7.4	Crafting knowledge from data	118
7.5	Knowledge-guided machine learning: reflections and future directions	120
7.5.1	Model evaluation	120
7.5.2	Regularization	120
7.5.3	Generalization and transferability	121
7.5.4	Model interpretability	121
7.6	Conclusion	122

I

PROLOGUE

1 INTRODUCTION

Quoique notre esprit ne puisse pas comprendre en soi le rapport de causalité qui existe entre les phénomènes et la matière, la science exige que nous admettions la nécessité d'un déterminisme dans cette causalité.

— Claude Bernard (1867)

Throughout history, mankind has sought to understand the inner workings of the world with different frameworks of explanation. The religious, at least the (mono-)theistic ones and in particular Christians, believe in the existence of an almighty God, architect and sustainer of the universe (Aquinas, 1947, ST I qq. 44-49), whose will manifests in all natural phenomena through *providence* (Aquinas, 1947, ST I q.22), filling them with purpose and order. Later on, the founders of modern science such as Newton (Newton, 1999), although religious and convinced of God's influence in the design of the world (Newton, 1999, p. 940), moved the explanation of why things happen from providential causation towards a law-based order supported by empirical observations (Funkenstein, 1986). By doing so, he confined rational explanations to a precise mathematical description. Building upon the observations and experiments of Galileo and Kepler about astronomy and planetary motion, Newton derived a framework providing a mechanical explanation for the entire universe, which we refer to as *model*. Although the earliest instances of mathematical models originated in Mesopotamia and Ancient Egypt, as demonstrated for example with the Rhind Mathematical Papyrus providing formulae to compute areas and volumes for different figures and geometries, Newton's laws constitute the first mathematically unified model representing abstract, theoretical physics about the world.

This most elegant system of the sun, planets, and comets could not have arisen without the design and dominion of an intelligent and powerful being. And if the fixed stars are the centers of similar systems, they will all be constructed according to a similar design and subject to the dominion of One.

— Isaac Newton (1687)

In its purest form, a model is defined as “a simple representation of a system [...] that can be used in [...] predictions of what might happen”¹. Therefore, it can be applied to almost any domain in which concepts and interactions can be transformed into mathematical abstractions and operations. Following Newton's theory, the Second Industrial Revolution was characterized by a surge in the use of scientific models in a variety of fields, from metallurgy to thermodynamics. For example, Sadi Carnot conceptualized the Carnot cycle in 1824, defining the theoretical maximum efficiency of traditional thermodynamic engines when converting thermal energy into work. Its principles led Rudolf Clausius to the derivation of the Second Law of Thermodynamics and served as blueprint to design novel steam turbines. A bit later, in the field of electromagnetism, James Clerk Maxwell derived a foundational mathematical framework in 1860, unifying electricity, optics and magnetism with a set of four equations. He showed that a change in the magnetic field induces the creation of an electric field, and conversely. His work transformed the manufacturing domain by connecting factories to power grids rather than traditional steam engines, and he also inspired Einstein's subsequent theories on special and general relativity. Such fundamental mathematical models were thus at the basis of both economical and societal growth. With the advent

¹Cambridge Dictionary (<https://dictionary.cambridge.org/dictionary/english/model>) - Accessed: 2026-07-02

of computers, mathematical models could be digitally implemented as surrogates for many complex (industrial) processes; yet their efficient applicability remained limited. For example, solving models described by partial differential equations (PDE) using the finite element method (FEM) (Zienkiewicz et al., 2000) may prove expensive, due to large computational demands induced by high-quality meshes. Hence, practitioners should carefully design preliminary studies to decide, for instance, on the most relevant set of control parameters to be tested for simulations of a given process.

In the 1950s, the premises of machine learning (ML) and artificial intelligence came to life with Alan Turing’s work on the subject (Turing, 1950). In the objective of mimicking physiological and biological processes, Frank Rosenblatt invented the very first models of neurons and the (multi-layer) perceptron (more formerly known as neural network) in 1957 (Rosenblatt, 1958). In 1986, Rumelhart et al. (1986) formalized the backpropagation algorithm, providing a generic mathematical recipe to train the weights of said neural networks, thereby paving the way for modern deep learning research. Indeed, this breakthrough notably allowed the broader practical design and training of more and more complex architectures, from early convolutional neural networks (W. Zhang et al., 1988; LeCun et al., 1989) and recurrent neural networks (Elman, 1990) to more recent generative methods such as diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) and large language models (Brown et al., 2020). All of a sudden, machines are able to automatically discover new models and patterns in unorganized quantities of data from which domain experts were not able to derive any physical nor mathematical sense. Given pairs of input features and output values, models approximating the true input-to-output relationship start to emerge, which we call *data-driven* models. Recently, several works addressed the problem of modeling trajectories simulated from given sets of PDEs (Chen et al., 2018; Li et al., 2020; Biloš et al., 2021) with data-driven models, which allows the efficient generation of new trajectories, as opposed to traditional time-expensive FEM-based solvers.

Yet, the crucial condition for the successful deployment of such models remains the availability of abundant training data. For certain applications like text generative modeling tasks, it is now very easy to collect lots of data by crawling the web. A very well-known example of large dataset used for benchmarking purposes in image classification tasks is the ImageNet dataset (Deng et al., 2009), which consists of over a million training images with their associated target label, covering a total of 1 000 classes. With infinite training data, traditional learning algorithms should in principle yield an optimal model, called the *Bayes* model. Still, there exist tasks and domains for which data is not as easily accessible, such as healthcare, or at least for which there do not exist curated high-quality datasets large enough to provide sufficient information about the true data distribution (Shaikhina et al., 2015; Razzak et al., 2017; Bansal et al., 2022; Upadhyay et al., 2024). Taking back the previous example about FEM-generated trajectories of PDEs, it could take a very important amount of time to constitute a decent database, covering a sufficiently wide span of initial and boundary conditions. Furthermore, a data-driven model could be trained to reproduce trajectories for a given configuration of physical parameters, but it would certainly not generalize to different values of those parameters. With only limited available data, the predictive performance of the model and its robustness to unseen conditions are to be questioned, as ML models in general are prone to overfitting. This is even more true (or even worse) when the number of collected samples is smaller than the number of input features. One can imagine training deep learning models with small databases for research purposes, yet one can hardly imagine deploying such models for practical applications.

Another challenge inherent to ML models is the opaqueness to understanding their decision process easily. In fact, they are often referred to as “black boxes”, in opposition to “white boxes”, i.e., models driven explicitly by known principles and dynamics.

If a machine is expected to be infallible, it cannot also be intelligent.

— Alan Turing (1947)

Unfortunately, white-box models tend to be too simplistic as they can ignore parts of the underlying physics, thereby leading to important modeling errors. Although there exist ML models that are interpretable by design, e.g., single decision trees, linear models or more advanced versions of the latter such as generalized linear models (GLM) (Nelder et al., 1972) or generalized additive models (GAM) (Hastie et al., 1986), their predictive performance remains limited in comparison with the expressive capacity of neural networks which are considered universal approximators (Hornik et al., 1989). Recently, the field of explainable artificial intelligence (XAI) has vastly expanded (Adadi et al., 2018; Gilpin et al., 2018; Linardatos et al., 2020) and several post-hoc interpretation techniques have been developed to try to understand the behavior of black-box models. While multiple methods were proposed to measure the importance of features, e.g., SHAP values (Lundberg et al., 2017) and LIME (Ribeiro et al., 2016), other visual interpretation techniques such as partial dependence plots (Friedman, 2001), individual conditional expectation plots (Goldstein et al., 2015) or accumulated local effects plots (Apley et al., 2020) have also been developed to represent model responses to particular features of interest. Yet, such methods only help to understand how decisions are made, but they cannot correct the model if the decision was wrong, nor if it was right but for the wrong reasons.

To overcome the aforementioned issues of data scarcity and model opaqueness, knowledge-guided machine learning (KGML) approaches have been introduced. Such methods incorporate *prior* domain knowledge within the learning process of ML models, and knowledge can be used in various forms (Von Rueden et al., 2021; Willard et al., 2022). A common hybrid modeling approach consists in summing learnable first-principles and ML models, which structurally enforces the inductive bias to somehow respect known (approximate) physics. While this type of technique has proven useful to train more data-efficient models (Yin et al., 2021; Mehta et al., 2021; Donà et al., 2022), finding the right balance between both models remains challenging, if the goal is also to retrieve a physically plausible estimation of the first-principles model. To improve the reasoning behind decisions of (hybrid) models, several techniques have been developed as part of the field of explanation-guided learning (EGL) (Gao, Gu, et al., 2024), a domain whose goal is to align model explanations with expert knowledge. For example, prior knowledge about feature importance scores can be used to improve the performance of classifiers (Rieger et al., 2020; Balayan et al., 2020) by favoring important features and interactions while penalizing unimportant ones. In image processing tasks, user information about regions of higher importance can also be leveraged (Mitsuhara et al., 2019; Gao, Sun, et al., 2022; Y. Zhang et al., 2024).

1.1 RESEARCH QUESTION

Motivated by the continuous developments in the field of knowledge-guided machine learning and the existing challenges of physical plausibility and interpretability, this thesis focuses on developing novel learning processes for both hybrid models and models with enhanced interpretability, in the particular context of supervised regression.

The central part of this thesis is the concept of knowledge. In simple terms, knowledge can be seen as the understanding one has about a given subject and can therefore take many forms depending on the level of understanding and its nature. For example, it can consist in defining a tangible parametric partial model as it can also correspond to more abstract notions such as the satisfaction of some mathematical properties. Settling to a particular type of knowledge implies defining a given set of assumptions, some of which being more limiting. Therefore, this thesis tries to use knowledge of various forms to obtain ML models that are inherently more robust, more interpretable and more data-efficient. More formally, it studies the following research questions:

- (RQ1) Given prior knowledge in the form of a partial model, how can we design training methods that produce optimal hybrid additive models, both in terms of predictive performance as in the estimation of the parametric prior?
- (RQ2) How can the additivity assumption of hybrid models be relaxed by leveraging prior knowledge in the form of model explanations?
- (RQ3) How can partial auxiliary datasets be considered as prior knowledge in order to improve the predictive performance of machine learning models in the context of data scarcity?

1.2 OUTLINE

After this introduction, we list in Section 1.3 the different contributions of this thesis, which emerge from different scientific publications. Then, Chapter 2 provides some context about the vast field of knowledge-guided machine learning, covering related works that incorporate prior knowledge of various forms with machine learning methods, let it be in the architectural design of the framework, in the training process, but also to enhance the understanding of the decision process in general. This thesis focuses on two forms of prior knowledge, namely in the form of a partial mathematical model or in the form of a partial dataset. Therefore, Chapter 3 introduces a novel algorithm to train the first-principles component of a hybrid additive model combining partial knowledge, in the form of a parametric model, with a machine learning model serving as residual (RQ1). It also investigates the use of traditional tree-based machine learning methods in such hybrid frameworks, on a range of supervised regression problems. We then extend this algorithm in Chapter 4 to tackle the problem of dynamical systems forecasting with hybrid additive models. In Chapter 5, we address the inherent interpretability issues of neural networks by developing a training algorithm that leverages prior knowledge in the form of a partial dependence function about the final decision process (RQ2). We also bridge the gap between physics-informed machine learning and explanation-guided learning by comparing both methodologies on several performance indicators. Chapter 6 tackles the problem of transferring knowledge from larger yet feature-incomplete databases to models learned on much smaller datasets (RQ3). Finally, Chapter 7 concludes on the different findings presented in this thesis and discusses potential future directions in the field of knowledge-guided machine learning.

1.3 PUBLICATIONS

The main content of this thesis stems from the publication of the following peer-reviewed scientific works.

Yann Claes, Vân Anh Huynh-Thu, and Pierre Geurts (2023). “Knowledge-guided additive modeling for supervised regression”. In *International Conference on Discovery Science*. Springer, pp. 64–78,

Yann Claes, Vân Anh Huynh-Thu, and Pierre Geurts (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72,

Yann Claes, Pierre Geurts, and Vân Anh Huynh-Thu (2026). “Steering Neural Network Training through Interpretable Constraints Based on Partial Dependence”. In *arXiv preprint arXiv:2607.08641*. Submitted to TMLR.

Yann Claes, Pierre Geurts, and Vân Anh Huynh-Thu (2026). “Learning with Small Data by Transferring Marginal Effects from Large Data”. To appear in proceedings of

International Workshop on Learning from Small Data - ECML-PKDD.

During my thesis, I had the chance to collaborate on different projects, leading to the following publications, which are not featured in the manuscript.

Gaspard Lambrechts, Yann Claes, Pierre Geurts, and Damien Ernst (2024). “Parallelizing Autoregressive Generation with Variational State Space Models”. In *arXiv preprint arXiv:2407.08415*.

Florent De Geeter, Yann Claes, Vân Anh Huynh-Thu, Victor Dachet, Lize Pirenne, Omer Rochman Sharabi, Tojo Valisoa Andrianandrianina Johanesa, Antoine Gratia, Lucile Dierckx, Larbi Arbaoui, et al. (2024). “Temperature Field Prediction in Additive Manufacturing Process using Machine Learning”. In *The 13th ASMO UK/2nd ASMO-EUROPE/ISSMO conference on Engineering Design Optimization Product and Process Improvement*.

REFERENCES

- Adadi, Amina and Mohammed Berrada (2018). “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)”. In *IEEE access* 6, pp. 52138–52160.
- Apley, Daniel W and Jingyu Zhu (2020). “Visualizing the effects of predictor variables in black box supervised learning models”. In *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.4, pp. 1059–1086.
- Aquinas, Thomas (1947). “Summa Theologiae”. Translated by the Fathers of the English Dominican Province. Prima Pars, qq. 22, 44–49, 103–105. Benziger Brothers.
- Balayan, Vladimir et al. (2020). “Teaching the machine to explain itself using domain knowledge”. In *arXiv preprint arXiv:2012.01932*.
- Bansal, Ms Aayushi, Dr Rewa Sharma, and Dr Mamta Kathuria (2022). “A systematic review on data scarcity problem in deep learning: solution and applications”. In *ACM Computing Surveys (Csur)* 54.10s, pp. 1–29.
- Bernard, Claude (1867). “Principes de médecine expérimentale”. Emile Martinet.
- Biloš, Marin et al. (2021). “Neural flows: Efficient alternative to neural ODEs”. In *Advances in neural information processing systems* 34, pp. 21325–21337.
- Brown, Tom et al. (2020). “Language models are few-shot learners”. In *Advances in neural information processing systems* 33, pp. 1877–1901.
- Chen, Ricky T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In *Advances in Neural Information Processing Systems*.
- Claes, Yann, Pierre Geurts, and Vân Anh Huynh-Thu (2026). “Steering Neural Network Training through Interpretable Constraints Based on Partial Dependence”. In *arXiv preprint arXiv:2607.08641*. Submitted to TMLR.
- Claes, Yann, Vân Anh Huynh-Thu, and Pierre Geurts (2023). “Knowledge-guided additive modeling for supervised regression”. In *International Conference on Discovery Science*. Springer, pp. 64–78.
- (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.
- De Geeter, Florent et al. (2024). “Temperature Field Prediction in Additive Manufacturing Process using Machine Learning”. In *The 13th ASMO UK/2nd ASMO-EUROPE/ISSMO conference on Engineering Design Optimization Product and Process Improvement*.
- Deng, Jia et al. (2009). “Imagenet: A large-scale hierarchical image database”. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Elman, Jeffrey L (1990). “Finding structure in time”. In *Cognitive science* 14.2, pp. 179–211.
- Friedman, Jerome H (2001). “Greedy function approximation: a gradient boosting machine”. In *Annals of statistics*, pp. 1189–1232.
- Funkenstein, Amos (1986). “Theology and the Scientific Imagination: From the Middle Ages to the Seventeenth Century, Second Edition”. 2nd ed. Princeton University Press.
- Gao, Yuyang, Siyi Gu, et al. (2024). “Going beyond xai: A systematic survey for explanation-guided learning”. In *ACM Computing Surveys* 56.7, pp. 1–39.

- Gao, Yuyang, Tong Steven Sun, et al. (2022). "Aligning eyes between humans and deep neural network through interactive attention alignment". In *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–28.
- Gilpin, Leilani H et al. (2018). "Explaining explanations: An overview of interpretability of machine learning". In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, pp. 80–89.
- Goldstein, Alex et al. (2015). "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation". In *journal of Computational and Graphical Statistics* 24.1, pp. 44–65.
- Hastie, Trevor and Robert Tibshirani (1986). "Generalized additive models". In *Statistical science* 1.3, pp. 297–310.
- Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models". In *Advances in neural information processing systems* 33, pp. 6840–6851.
- Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In *Neural networks* 2.5, pp. 359–366.
- Lambrechts, Gaspard et al. (2024). "Parallelizing Autoregressive Generation with Variational State Space Models". In *arXiv preprint arXiv:2407.08415*.
- LeCun, Yann et al. (1989). "Backpropagation applied to handwritten zip code recognition". In *Neural computation* 1.4, pp. 541–551.
- Li, Zongyi et al. (2020). "Fourier neural operator for parametric partial differential equations". In *arXiv preprint arXiv:2010.08895*.
- Linardatos, Pantelis, Vasilis Papastefanopoulos, and Sotiris Kotsiantis (2020). "Explainable ai: A review of machine learning interpretability methods". In *Entropy* 23.1, p. 18.
- Lundberg, Scott M and Su-In Lee (2017). "A unified approach to interpreting model predictions". In *Advances in neural information processing systems* 30.
- Mehta, Viraj et al. (2021). "Neural Dynamical Systems: Balancing Structure and Flexibility in Physical Prediction". In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 3735–3742.
- Mitsuhara, Masahiro et al. (2019). "Embedding human knowledge into deep neural network via attention map". In *arXiv preprint arXiv:1905.03540*.
- Nelder, John Ashworth and Robert WM Wedderburn (1972). "Generalized linear models". In *Journal of the Royal Statistical Society Series A: Statistics in Society* 135.3, pp. 370–384.
- Newton, Isaac (1999). "The Principia: Mathematical Principles of Natural Philosophy". Trans. by I. Bernard Cohen and Anne Whitman. Originally published 1687; General Scholium added in the 2nd edition, 1713. University of California Press.
- Razzak, Muhammad Imran, Saeeda Naz, and Ahmad Zaib (2017). "Deep learning for medical image processing: Overview, challenges and the future". In *Classification in BioApps: Automation of decision making*, pp. 323–350.
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). "Why should i trust you?" Explaining the predictions of any classifier". In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.
- Rieger, Laura et al. (2020). "Interpretations are useful: penalizing explanations to align neural networks with prior knowledge". In *International conference on machine learning*. PMLR, pp. 8116–8126.
- Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In *Psychological review* 65.6, p. 386.

- Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In *nature* 323.6088, pp. 533–536.
- Shaikhina, Torgyn et al. (2015). "Machine learning for predictive modelling based on small data in biomedical engineering". In *IFAC-PapersOnLine* 48.20, pp. 469–474.
- Sohl-Dickstein, Jascha et al. (2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In *International conference on machine learning*. pmlr, pp. 2256–2265.
- Turing, A. M. (1950). "Computing Machinery and Intelligence". English. In *Mind*. New Series 59.236, pp. 433–460.
- Upadhyay, Ashwini Kumar and Ashish Kumar Bhandari (2024). "Advances in deep learning models for resolving medical image segmentation data scarcity problem: a topical review". In *Archives of Computational Methods in Engineering* 31.3, pp. 1701–1719.
- Von Rueden, Laura et al. (2021). "Informed Machine Learning—A taxonomy and survey of integrating prior knowledge into learning systems". In *IEEE Transactions on Knowledge and Data Engineering* 35.1, pp. 614–633.
- Willard, Jared et al. (2022). "Integrating scientific knowledge with machine learning for engineering and environmental systems". In *ACM Computing Surveys* 55.4, pp. 1–37.
- Yin, Yuan et al. (2021). "Augmenting physical models with deep networks for complex dynamics forecasting". In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.
- Zhang, Wei et al. (1988). "Shift-invariant pattern recognition neural network and its optical architecture". In *Proceedings of annual conference of the Japan Society of Applied Physics*. Vol. 564. Montreal, CA, p. 734.
- Zhang, Yifei et al. (2024). "Megl: Multimodal explanation-guided learning". In *arXiv preprint arXiv:2411.13053*.
- Zienkiewicz, Olgierd Cecil et al. (2000). "The finite element method". Elsevier.

II

MATTER

2 RELATED WORK

Prior knowledge about inference problems, either physical or purely mathematical, can guide choices in designing new machine learning frameworks with enhanced generalization capabilities, as it can also yield more interpretable and more data-efficient models. In this chapter, we categorize different knowledge integration mechanisms and review some notable works to give a general shape of the broad field of knowledge-guided machine learning research. Section 2.1.1 reviews works introducing prior knowledge as regularization constraints in the training loss, whereas in Section 2.1.2, we present several ways of integrating knowledge in the form of a partial model in the prediction of the model. Section 2.1.3 covers physics-informed machine learning techniques that incorporate prior knowledge in a less structured manner. Afterwards, Section 2.2 presents different techniques of explanation-guided learning, where prior knowledge about model explanations guides model training. We review techniques which leverage local (instance-level) model explanations in Section 2.2.1, whereas Section 2.2.2 covers methods that steer model training using information about global explanations. Finally, Section 2.3 positions our work within the different presented fields.

Contents

2.1	Knowledge-guided machine learning	12
2.1.1	Physics-informed training loss functions	12
2.1.2	Prior knowledge as hard constraints	12
2.1.3	Physics-informed machine learning architectures	15
2.2	Explanation-guided learning	17
2.2.1	Local explanation alignment	17
2.2.2	Global explanation alignment	18
2.3	On the concept of knowledge	18

2.1 KNOWLEDGE-GUIDED MACHINE LEARNING

In this section, we provide an overview which categorizes several knowledge-guided machine learning techniques into distinct (sub-)families of approaches. In particular, Section 2.1.1 covers works that specifically incorporate prior (physical) knowledge as regularization terms in the training loss function, whereas Section 2.1.2 reviews works that introduce prior knowledge as a structural part of the predictive model, e.g., through partially-known parametric models. Finally, we include in Section 2.1.3 frameworks whose implementation has been strongly influenced by specific prior knowledge about the problem at hand, but which do not fall into the two previous categories. This can be, for instance: by exploiting physical analogies, by feeding the outputs of prior models to statistical models in fuzzy ways (and vice versa), but also by entangling models with physical meaning, let it be through the prediction of physical parameters or through the physical quantity they (supposedly) represent.

2.1.1 PHYSICS-INFORMED TRAINING LOSS FUNCTIONS

Prior knowledge can be used to guide model training in a form of scientific regularization, by incorporating additional discrepancy/similarity terms to the training loss function, which encourages models to respect the properties at hand.

Physical properties can be induced in supervised regression problems, namely for process modeling tasks. For instance, to tackle the problem of lake temperature modeling, Jia et al. (2021) compute the total thermal energy of the lake from the predicted output temperature and enforce conservation of the total energy by minimizing the mean-squared error w.r.t. the difference of incoming/outgoing heat fluxes. A similar idea can be found in (S. Chen et al., 2024), where expert knowledge about optical flows and directions of fracture development helped shape additional regularization terms. Similar principles can also be applied with other varieties of neural networks, such as graph neural networks, as done in (Seo et al., 2019), where authors encode physical properties (both static and dynamic, e.g., diffusion) with appropriate node and edge update functions that are then included to the loss function for minimization of the residuals. In generative modeling tasks, Zeng et al. (2021) implement imprecise physical constraints as additional loss terms to the generator model of a generative adversarial model, which helps training the latter in cases where the discriminator has already converged.

Aside strictly physical properties, other mathematical and generic properties can serve as regularizing functions. For example, Muralidhar et al. (2018) simply define a constraining term with a ReLU unit to penalize non-monotonicity in the input-output relationship. More generally, Beucler et al. (2021) define a fixed constraint matrix that is applied on the output of neural networks to enforce a predefined set of required properties.

Finally, loss functions can also carry purely physical meaning. Stewart et al. (2017) assume a configuration where no target labels are available for traditional supervised regression. Instead, for object tracking problems, given a sequence of input images, they define a loss function only from physical properties that should be verified by the object in motion, e.g., the trajectory of an object in free fall, or trajectory under constant velocity.

2.1.2 PRIOR KNOWLEDGE AS HARD CONSTRAINTS

Scientific information can be available as partial knowledge about the target input-output relationship, which can take the form of constraints or logical rules defined over given sub-spaces of the original domain, but also as partial parametric information about the model itself, which is often called a first-principles model. In this section, we synthesize some related works covering the inclusion of such specific knowledge, which we refer to

as *hard constraints* in the sense that they are encoded in the inductive bias of the model by construction.

LEVERAGING DOMAIN CONSTRAINTS AND LOGICAL RULES

For some problems, only partial knowledge about the target model is available. This can be for example through constraints that should be satisfied in some regions of the input space. For classification problems, both Fung et al. (2002) and Mangasarian et al. (2008) reformulate the support vector machine linear problem to include knowledge constraints about the domain. Lauer et al. (2008) use similar knowledge to enhance the performance of support vector machines applied to regression problems. Yang et al. (2019) define constraint priors over network weights for Bayesian neural networks, which consist of information about where outputs given certain inputs can or cannot lie. For tree-based models, Nanfack et al. (2023) derive a new tree formulation based on matrices representing branching decisions and show that these matrices can easily be constrained to satisfy prior domain constraints defined on the branching process. Regarding traditional neural networks, Goyal et al. (2024) design a training approach which guarantees that domain constraints are satisfied by model outputs, based on maximum satisfiability theory. Theoretically, Tulabandhula et al. (2015) formalize the guarantees obtained when injecting such knowledge about domain constraints and show that it helps generalization.

ENCODING PHYSICAL PROPERTIES INTO MACHINE LEARNING MODELS

Prior knowledge about the problem at hand can be directly incorporated in the implementation of specific parts of machine learning models. For example, a domain expert could inform the practitioner that the relationship between a specific subset of features and the target variable should be monotonic. To address this requirement, Wehenkel et al. (2019) reframe the prediction problem by instead modeling the derivative of the target function by an unconstrained neural network, whose output is first passed through an ELU activation incremented by 1. This approach imposes minimal restrictions to the neural network while ensuring monotonicity through the strictly positive derivative. Pya et al. (2015) design a machine learning architecture that imposes properties such as monotonicity or convexity by implementing a structure similar to generalized additive models (Hastie et al., 1986) where the corresponding predicting terms have a shape that is constrained by the construction of a certain splines basis. Riihimäki et al. (2010) introduce the concept of virtual derivatives to incorporate monotonicity information in multivariate Gaussian processes.

From the angle of applied machine learning, for the particular problem of lake temperature modeling, Daw, Thomas, et al. (2020) introduce an additional modeling step to approximate water density, whose relationship given depth is known to be monotonic. Therefore, they complement an LSTM cell with appropriate operations to ensure that monotonicity is preserved between water depth (given as input) and density, and eventually estimate lake temperature from the original inputs and the predicted density, with a standard black-box model.

DESIGNING HYBRID ADDITIVE MODELS

For other problems, one has access to a partial model definition, for example describing incomplete physical knowledge about the system at hand. A natural approach in this case is to enhance first-principles models by predicting the residual between the observed and physical outputs. Historically, machine learning models have been additively combined with physics-based models to solve process modeling tasks, either with radial basis functions (Kramer et al., 1992) or with traditional neural networks (Su et al., 1993).

Even though the seminal work of Thompson et al. (1994) studies parallel and serial combinations of both types of model, we will mostly restrict to additive combinations of first-principles and machine learning models, as considered in (Duarte et al., 2004; D. S. Lee et al., 2005).

Other works consider weaker prior knowledge, e.g., in the form of linear models. For instance, J. Chen et al. (2007) additively combine the effects of a linear model and a tree-based model through an iterative training procedure. H. Zhang et al. (2019) additively combine a trained linear model with a random forest regressor fitted on the residual, and show that this hybrid structure improves generalization in several extrapolation settings.

Yet, with no further constraints in the training process, such additive framework can often have multiple solutions that are valid in the sense of least-squares minimization of the global model, but for which the physical meaning of the first-principles model is completely lost. This could likely happen in cases where the statistical model, e.g., a neural network known to be a universal approximator, captures the entire signal thereby forcing the physical model to zero. These questions about the uniqueness of the solution motivate several works that focus on methods to find the right balance between both components.

REGULARIZING HYBRID ADDITIVE MODELS

Guen et al. (2020) tackle the problem of video prediction by additively disentangling physical dynamics in latent space as a sum $\mathcal{M}_p + \mathcal{M}_r$, where $\mathcal{M}_p$ is a generic convolutional cell that encodes a set of differential equations and $\mathcal{M}_r$ is recurrent neural network that encodes residual dynamics. To enhance the meaningfulness of the first-principles model, they impose moment regularization constraints on the learned filters of the cell such that it behaves as a PDE solver; yet those do not strictly prevent the machine learning model from absorbing the whole signal, although discouraged if $\mathcal{M}_p$ captures the dynamics quite well. Djeumou et al. (2022) design a hybrid model whose decomposition is not limited to additive models, as the unknown dynamics function could be integrated within the partial physics model similarly to (Thompson et al., 1994), or multiplied with the latter, as is done for the double pendulum problem. Nevertheless, they assume perfect knowledge of the physical prior and hence only regularize the statistical component with constraints such as symmetries, to improve the physical significance of the hybrid model.

Yin et al. (2021) develop APHYNITY, a hybrid additive model training approach whose first-principles term F_p represents partial model knowledge about the dynamical process, e.g., for a damped pendulum system, F_p could be the periodic motion function, omitting the additive friction dynamics. Compared to Guen et al. (2020), F_p structurally encodes the right physics, with learnable parameters. Therefore, it is crucial to design an appropriate training scheme to ensure that the physical parameters remain physically plausible. For that purpose, they introduce a regularization term on the machine learning term F_a to keep its contribution to the global dynamics as minimal as possible, by minimizing its $\mathcal{L}^2$ -norm. In a similar line of work, Donà et al. (2022) propose the same additive decomposition as APHYNITY for dynamical systems forecasting tasks, but the training optimization scheme differs. While models in the latter are optimized jointly, they alternate training steps between F_p and F_a (on current residuals) and introduce a regularization term for F_p to keep it close to its initial approximation, whose importance decays over time. In generative modeling tasks, Takeishi et al. (2021) derive physics-integrated variational auto-encoders, where the latent space is divided into vectors z_p and z_a that are eventually summed before decoding. z_p is mapped to the physical parameter vector of the first-principles model, and further intricate regularization terms ought to ground it to the true physics.

As z_p should correspond in meaning to some physical parameters, their method allows handling different values of such parameters without retraining.

2.1.3 PHYSICS-INFORMED MACHINE LEARNING ARCHITECTURES

In this section, we review broader forms of physics-informed machine learning architectures, which cannot simply be categorized into the previous two families of approaches. This includes, among others: predictive frameworks where physics serves as methodological analogies, original combinations of physical priors and machine learning outputs (different from pure summation), as well as machine learning models that carry a directly interpretable physical meaning.

WHEN PHYSICS INFLUENCES MODELING PIPELINES

Physical analogies as methodological guidelines The design of traditional machine learning predictive pipelines can be modified to respect known physical principles by construction, or even reframed based on physical analogies. For instance, He et al. (2020) develop a fluid-inspired paradigm to solve point-cloud predictive tasks, where points are transformed into feature particles on a grid and whose evolution is implemented through advective field prediction. Another example of physical analogy can be found in (Ji et al., 2022), where traffic flow prediction is framed as an energy transport problem, by assuming that traffic flow is an edge feature of a road graph driven by differences of potential energy fields representing nodal road junctions. Finally, physical relationships can also guide the practical conception of specific predictive cells, as is done in (L. Liu et al., 2022) where gated recurrent unit cells are organized such that the information flow of the global architecture follows some causal order, for the process of nitrous oxide flux prediction.

Post-processing model outputs with physics Besides frameworks resulting from physical parallels, first-principles and statistical models can be combined in various ways, influenced by prior knowledge about the problem at hand. For instance, machine learning models can carry some substantial meaning that can then be exploited by physically-grounded models (or solvers) to improve predictive performance. In (Kochkov et al., 2021), the filters of a convolutional neural network are constrained by physical properties and its outputs, representing latent velocity fields, are eventually post-processed by specific physical blocks such as the divergence operator to yield new velocities. De Bézenac et al. (2019) use an encoder-decoder model to predict the motion field from a series of input images, from which the last image is then physically warped according to the estimated motion field. In (Holl et al., 2020), a combination of two neural networks is designed to predict approximate trajectories with appropriate corrections to reach a certain target observation, both of them being then fed into a differentiable solver that explicitly encodes physical rules to ensure physical consistency.

Refining physical approximations with machine learning models Conversely, the output of an approximate prior model can be fed as additional input feature to a machine learning model, which acts as a corrector. For example, in (Daw, Karpatne, et al., 2022), a neural network takes as input the output of a physical simulator modeling a lake temperature process, to produce a refined temperature prediction. In the context of dynamical processes, Tathawadekar et al. (2023) leverage an physically-incomplete PDE solver to obtain a starting trajectory, which is then fed to a machine learning model that produces appropriate corrections to each state. For the more generic problem of flow simulation, the coarse output simulated by a computational fluid dynamics model can be fed to graph neural networks (Belbute-Peres et al., 2020; R. Zhao et al., 2024) or U-Net architectures (Pathak et al., 2020) for further refinement.

Physics as conditional guidance for generative modeling Lastly, in generative modeling tasks, prior knowledge can serve as conditional guidance. For instance, Shu et al. (2023) enhance the problem of high-fidelity flow field reconstruction by using the gradient of the PDE residual estimated from the generated solution, at each time step, as conditional training guidance. As the previous solution conditions a model only on a given set of PDEs, (Jacobsen et al., 2025) solve this issue by computing such gradient at inference time and by adding it to the predicted score. More generic properties like energy conservation can also be incentivized by multiplying the predicted likelihood with a discrepancy weight that penalizes physically unsound solutions (Z. Gao et al., 2023).

GROUNDING NEURAL NETWORKS TO PHYSICAL QUANTITIES

Machine learning models can be formalized so as to directly represent meaningful scientific quantities. Historically, Psychogios et al. (1992) combine a neural network with a first-principles model in a serial approach, for the application of fedbatch bioreactor modeling. In particular, the neural network is responsible for the prediction of the cell growth rate given process parameters, and this predicted rate is then substituted into the expression of the first-principles model. More recently, Ling, Kurzawski, et al. (2016) used neural networks to model Reynolds stresses, with appropriate physical post-processing to embed Galilean invariance into the predicted stress tensors. A similar approach was developed using Random Forests (Ling, Ruiz, et al., 2017), yet with no possibility to enforce theoretical properties. Shen et al. (2023) implement (differentiable) geoscientific physical processes directly in PyTorch (Paszke et al., 2019) and rely on neural networks to predict the appropriate physical parameters that are given to the solver.

In a similar philosophy of entangling models with physical meaning, the differentiability of neural networks can be leveraged such that they obey the rules of given mechanics systems. A notable example is Symplectic ode-net (Zhong et al., 2019), a framework designed to enforce hamiltonian dynamics where three neural networks are used to represent respectively the external forcing, the inertia matrix and the potential energy, at given input coordinates. These networks are then explicitly combined within Hamiltonian dynamics, which are solved using Neural ODE (R. T. Q. Chen et al., 2018) to obtain the predicted trajectories. A similar modeling approach are the Deep Lagrangian Networks (Lutter et al., 2019), where two neural networks are used to explicitly represent the inertia matrix and gravitational forces in Lagrangian dynamics, given input coordinates. While neural networks constitute here a mapping to physical quantities involved in Hamiltonian/Lagrangian mechanics, another possibility can be to bound them directly to the central objects of these dynamics. Hence, Greydanus et al. (2019) model the Hamiltonian through a neural network and rely on automatic differentiation to fit its gradients to target derivative values. Cranmer et al. (2020) employ the same approach for Lagrangian mechanics. Finally, network gradients can be optimized in a fully unsupervised manner such that the model acts as a solution to a particular PDE system. In their seminal work, Lagaris et al. (1998) assume knowledge of the shape of the PDE and design a specific architecture for its solution which enforces boundary conditions by construction. Given that gradients must be computed analytically, they restrict to simple structures for the neural network, whose gradients are optimized to minimize the PDE residual. More recently, Raissi et al. (2019) leverage modern computation to minimize the residual with automatic differentiation, which also allows to enforce arbitrary boundary conditions, thereby giving more flexibility in the design of such kinds of models.

2.2 EXPLANATION-GUIDED LEARNING

While one can argue that the physics-informed approaches presented previously yield interpretable decisions, they nonetheless all rely on black-box models at some stage to produce the final prediction. For example, hybrid additive models have a stronger propensity to be more interpretable by construction but the residual component remains harder to explain naturally.

Recently, there has been a vast development of the field of explainable artificial intelligence (XAI), as a way to improve our understanding of machine learning models (Adadi et al., 2018; Gilpin et al., 2018; Linardatos et al., 2020). Although most of these tools can help domain experts to understand the decision process, they cannot take any feedback into account and hence cannot correct the model if it provides a correct decision for the wrong reasons, for instance.

Explanation-guided learning specifically addresses such shortcomings by trying to align model explanations with expert knowledge. In this section, we summarize important paradigms of the field, which we divide into two categories, namely local guidance, which leverages explanation information on a sample basis, and global guidance, whose goal is to align the general decision process of the model with prior knowledge.

2.2.1 LOCAL EXPLANATION ALIGNMENT

At a local level, the idea is to use prior explanation information available for given samples. In text classification tasks, the attention weights of the model can be supervised to put more importance to words that belong to the explanations, called rationales, a technique named rationale attention alignment (Ye Zhang et al., 2016; Kanchinadam et al., 2020; Stacey et al., 2022). A similar technique can be applied to image classification problems, based on saliency maps (Zhou et al., 2016), a process called visual alignment. For example, Y. Gao, Sun, Bai, et al. (2022) enhance model explainability via robust supervision of the model attention maps on potentially incomplete human annotations of target maps. The same idea is used in (Mitsuhara et al., 2019; Y. Gao, Sun, L. Zhao, et al., 2022), with the particularity that the model can be fine-tuned on problematic samples, i.e. samples for which the produced attention map is incorrect and for which humans can provide corrected versions. Yifei Zhang et al. (2024) combine both visual alignment and rationale alignment, feeding the feature maps and attention maps to a large language model for further alignment with textual explanations.

For more generic classification tasks, Rieger et al. (2020) allow the user to specify a priori feature importance vectors along with importance scores for interactions of features, so as to penalize unimportant features and favor relevant features in the final outcome. As such relevance scores can be too technical for domain experts to provide, Balayan et al. (2020) train a neural network to provide both a decision and a distribution over semantic concepts, the latter being verified by human experts to fine-tune model weights.

Finally, explanations can be exploited to guide model training without performing direct supervised training on target explanations. In the context of text classification, Zaidan et al. (2007) modify the traditional support vector machine training algorithm by integrating contrastive examples, crafted from original examples, such that there exists a clear difference in the obtained margins between positive examples and less positive (i.e. crafted) examples. Ismail et al. (2021) enforce the model to focus on the most important regions of the input by masking less relevant regions and adding a consistency loss term, so as to make model predictions for masked inputs close to those for unmasked inputs.

2.2.2 GLOBAL EXPLANATION ALIGNMENT

Guidance information can also be available to supervise the model’s global decision process. Erion et al. (2021) propose global attribution priors for higher-level properties such as sparsity, smoothness, or for example considering the relative importance between features instead of relying on fixed targets as done in (Rieger et al., 2020). These global attribution priors are meant to hold for all samples as they represent feature-wise explanations. In their work, F. Liu et al. (2019) mitigate bias in text classification tasks by penalizing non-zero attributions for keywords identifying protected groups. Weinberger et al. (2020) leverage meta-features to better align global importance values for each feature. Such meta-features include information indicating why certain features should be considered more important compared to others, for the given task.

Considering inherently interpretable models, Vojíř et al. (2020) design a rule-based classification system whose decision rules can later be modified by users through a specific web editor interface. More generally, in the context of active learning, Popordanoska et al. (2020) train a black-box classifier that is then distilled into an interpretable surrogate model. The latter provides global explanations in the form of rules along with decisions for unlabeled examples, which allow the domain expert to select relevant (mistaken) samples for further re-training. S. Lee et al. (2022) transform traditional deep learning models into self-explaining models which directly generates explanations, in the form of logical rules, by building the latter recursively from the model’s hidden representations, a process called *deep antecedent generation*. Specifically, after processing the input samples, the model first draws a rule from a learned distribution (conditioned on the input) and the prediction is then derived solely from that rule. This process can easily integrate global human priors about the form of rules to be generated.

2.3 ON THE CONCEPT OF KNOWLEDGE

While Section 2.1 covering knowledge-guided machine learning techniques is distinct from Section 2.2 which refers to explanation-guided learning approaches, both actually rely on the concept of knowledge for the implementation of modeling frameworks. However, what structurally separates both modeling classes is their usage of prior knowledge. In explanation-guided learning, as suggested by the naming, knowledge is leveraged in the form of explanations about the decision process, and such methods thus act on model explanations. On the contrary, in knowledge-guided machine learning, prior knowledge can be of various forms, but it constrains model outputs to be aligned.

In the following, we therefore position Chapters 3, 4 and 6 in the field of knowledge-guided machine learning as they directly affect model outputs, whereas Chapter 5 lies in the field of explanation-guided learning, through constraints on model explanations.

REFERENCES

- Adadi, Amina and Mohammed Berrada (2018). “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)”. In *IEEE access* 6, pp. 52138–52160.
- Balayan, Vladimir et al. (2020). “Teaching the machine to explain itself using domain knowledge”. In *arXiv preprint arXiv:2012.01932*.
- Belbute-Peres, Filipe De Avila, Thomas Economon, and Zico Kolter (2020). “Combining differentiable PDE solvers and graph neural networks for fluid flow prediction”. In *international conference on machine learning*. PMLR, pp. 2402–2411.
- Beucler, Tom et al. (2021). “Enforcing analytic constraints in neural networks emulating physical systems”. In *Physical review letters* 126.9, p. 098302.
- Chen, Jinbo et al. (2007). “A partially linear tree-based regression model for assessing complex joint gene–gene and gene–environment effects”. In *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* 31.3, pp. 238–251.
- Chen, Ricky T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In *Advances in Neural Information Processing Systems*.
- Chen, Shengyu et al. (2024). “HOSSNet: An efficient physics-guided neural network for simulating micro-crack propagation”. In *Computational Materials Science* 236, p. 112846.
- Cranmer, Miles et al. (2020). “Lagrangian neural networks”. In *arXiv preprint arXiv:2003.04630*.
- Daw, Arka, Anuj Karpatne, et al. (2022). “Physics-guided neural networks (pgnn): An application in lake temperature modeling”. In *Knowledge guided machine learning*. Chapman and Hall/CRC, pp. 353–372.
- Daw, Arka, R Quinn Thomas, et al. (2020). “Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling”. In *Proceedings of the 2020 siam international conference on data mining*. SIAM, pp. 532–540.
- De Bézenac, Emmanuel, Arthur Pajot, and Patrick Gallinari (2019). “Deep learning for physical processes: Incorporating prior scientific knowledge”. In *Journal of Statistical Mechanics: Theory and Experiment* 2019.12, p. 124009.
- Djeumou, Franck et al. (2022). “Neural networks with physics-informed architectures and constraints for dynamical systems modeling”. In *Learning for Dynamics and Control Conference*. PMLR, pp. 263–277.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Duarte, Belmiro, PM Saraiva, and CC Pantelides (2004). “Combined mechanistic and empirical modelling”. In *International Journal of Chemical Reactor Engineering* 2.1, pp. 1542–6580.
- Erion, Gabriel et al. (2021). “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”. In *Nature machine intelligence* 3.7, pp. 620–631.
- Fung, Glenn, Olvi Mangasarian, and Jude Shavlik (2002). “Knowledge-based support vector machine classifiers”. In *Advances in neural information processing systems* 15.
- Gao, Yuyang, Tong Steven Sun, Guangji Bai, et al. (2022). “Res: A robust framework for guiding visual explanation”. In *proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 432–442.

- Gao, Yuyang, Tong Steven Sun, Liang Zhao, et al. (2022). “Aligning eyes between humans and deep neural network through interactive attention alignment”. In *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–28.
- Gao, Zhihan et al. (2023). “Prediff: Precipitation nowcasting with latent diffusion models”. In *Advances in Neural Information Processing Systems* 36, pp. 78621–78656.
- Gilpin, Leilani H et al. (2018). “Explaining explanations: An overview of interpretability of machine learning”. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, pp. 80–89.
- Goyal, Kshitij, Sebastijan Dumancic, and Hendrik Blockeel (2024). “Deepsade: Learning neural networks that guarantee domain constraint satisfaction”. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 11, pp. 12199–12207.
- Greydanus, Samuel, Misko Dzamba, and Jason Yosinski (2019). “Hamiltonian neural networks”. In *Advances in neural information processing systems* 32.
- Guen, Vincent Le and Nicolas Thome (2020). “Disentangling physical dynamics from unknown factors for unsupervised video prediction”. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11474–11484.
- Hastie, Trevor and Robert Tibshirani (1986). “Generalized additive models”. In *Statistical science* 1.3, pp. 297–310.
- He, Xingzhe, Helen Lu Cao, and Bo Zhu (2020). “Advectivenet: An eulerian-lagrangian fluidic reservoir for point cloud processing”. In *arXiv preprint arXiv:2002.00118*.
- Holl, Philipp, Vladlen Koltun, and Nils Thuerey (2020). “Learning to control pdes with differentiable physics”. In *arXiv preprint arXiv:2001.07457*.
- Ismail, Aya Abdelsalam, Hector Corrada Bravo, and Soheil Feizi (2021). “Improving deep learning interpretability by saliency guided training”. In *Advances in Neural Information Processing Systems* 34, pp. 26726–26739.
- Jacobsen, Christian, Yilin Zhuang, and Karthik Duraisamy (2025). “Cocogen: Physically consistent and conditioned score-based generative models for forward and inverse problems”. In *SIAM Journal on Scientific Computing* 47.2, pp. C399–C425.
- Ji, Jiahao et al. (2022). “STDEN: Towards physics-guided neural networks for traffic flow prediction”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 36. 4, pp. 4048–4056.
- Jia, Xiaowei et al. (2021). “Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles”. In *ACM/IMS Transactions on Data Science* 2.3, pp. 1–26.
- Kanchinadam, Teja et al. (2020). “Rationale-based Human-in-the-Loop via Supervised Attention.” In *Dash@ kdd*. Association for Computing Machinery.
- Kochkov, Dmitrii et al. (2021). “Machine learning–accelerated computational fluid dynamics”. In *Proceedings of the National Academy of Sciences* 118.21, e2101784118.
- Kramer, Mark A, Michael L Thompson, and Phiroz M Bhagat (1992). “Embedding theoretical models in neural networks”. In *1992 American Control Conference*. IEEE, pp. 475–479.
- Lagaris, Isaac E, Aristidis Likas, and Dimitrios I Fotiadis (1998). “Artificial neural networks for solving ordinary and partial differential equations”. In *IEEE transactions on neural networks* 9.5, pp. 987–1000.
- Lauer, Fabien and Gérard Bloch (2008). “Incorporating prior knowledge in support vector regression”. In *Machine Learning* 70.1, pp. 89–118.

- Lee, Dae Sung, Peter A Vanrolleghem, and Jong Moon Park (2005). “Parallel hybrid modeling methods for a full-scale cokes wastewater treatment plant”. In *Journal of Biotechnology* 115.3, pp. 317–328.
- Lee, Seungeon et al. (2022). “Self-explaining deep models with logic rule reasoning”. In *Advances in neural information processing systems* 35, pp. 3203–3216.
- Linardatos, Pantelis, Vasilis Papastefanopoulos, and Sotiris Kotsiantis (2020). “Explainable ai: A review of machine learning interpretability methods”. In *Entropy* 23.1, p. 18.
- Ling, Julia, Andrew Kurzawski, and Jeremy Templeton (2016). “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance”. In *Journal of Fluid Mechanics* 807, pp. 155–166.
- Ling, Julia, Anthony Ruiz, et al. (2017). “Uncertainty analysis and data-driven model advances for a jet-in-crossflow”. In *Journal of Turbomachinery* 139.2, p. 021008.
- Liu, Frederick and Besim Avci (2019). “Incorporating priors with feature attribution on text classification”. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6274–6283.
- Liu, Licheng et al. (2022). “KGML-ag: a modeling framework of knowledge-guided machine learning to simulate agroecosystems: a case study of estimating N₂O emission using data from mesocosm experiments”. In *Geoscientific model development* 15.7, pp. 2839–2858.
- Lutter, Michael, Christian Ritter, and Jan Peters (2019). “Deep lagrangian networks: Using physics as model prior for deep learning”. In *arXiv preprint arXiv:1907.04490*.
- Mangasarian, Olvi L and Edward W Wild (2008). “Nonlinear knowledge-based classification”. In *IEEE Transactions on Neural Networks* 19.10, pp. 1826–1832.
- Mitsuhara, Masahiro et al. (2019). “Embedding human knowledge into deep neural network via attention map”. In *arXiv preprint arXiv:1905.03540*.
- Muralidhar, Nikhil et al. (2018). “Incorporating prior domain knowledge into deep neural networks”. In *2018 IEEE international conference on big data (big data)*. IEEE, pp. 36–45.
- Nanfack, Geraldin, Paul Temple, and Benoit Frenay (2023). “Learning customised decision trees for domain-knowledge constraints”. In *Pattern Recognition* 142, p. 109610.
- Paszke, Adam et al. (2019). “Pytorch: An imperative style, high-performance deep learning library”. In *Advances in neural information processing systems* 32.
- Pathak, Jaideep et al. (2020). “Using machine learning to augment coarse-grid computational fluid dynamics simulations”. In *arXiv preprint arXiv:2010.00072*.
- Popordanoska, Teodora, Mohit Kumar, and Stefano Teso (2020). “Machine guides, human supervises: Interactive learning with global explanations”. In *arXiv preprint arXiv:2009.09723*.
- Psichogios, Dimitris C and Lyle H Ungar (1992). “A hybrid neural network-first principles approach to process modeling”. In *AIChE Journal* 38.10, pp. 1499–1511.
- Pya, Natalya and Simon N Wood (2015). “Shape constrained additive models”. In *Statistics and computing* 25.3, pp. 543–559.
- Raissi, Maziar, Paris Perdikaris, and George E Karniadakis (2019). “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In *Journal of Computational physics* 378, pp. 686–707.
- Rieger, Laura et al. (2020). “Interpretations are useful: penalizing explanations to align neural networks with prior knowledge”. In *International conference on machine learning*. PMLR, pp. 8116–8126.

- Riihimäki, Jaakko and Aki Vehtari (2010). “Gaussian processes with monotonicity information”. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 645–652.
- Seo, Sungyong and Yan Liu (2019). “Differentiable physics-informed graph networks”. In *arXiv preprint arXiv:1902.02950*.
- Shen, Chaopeng et al. (2023). “Differentiable modelling to unify machine learning and physical models for geosciences”. In *Nature Reviews Earth & Environment* 4.8, pp. 552–567.
- Shu, Dule, Zijie Li, and Amir Barati Farimani (2023). “A physics-informed diffusion model for high-fidelity flow field reconstruction”. In *Journal of Computational Physics* 478, p. 111972.
- Stacey, Joe, Yonatan Belinkov, and Marek Rei (2022). “Supervising model attention with human explanations for robust natural language inference”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 36. 10, pp. 11349–11357.
- Stewart, Russell and Stefano Ermon (2017). “Label-free supervision of neural networks with physics and domain knowledge”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1.
- Su, Hong-Te et al. (1993). “Integrating neural networks with first principles models for dynamic modeling”. In *Dynamics and control of chemical reactors, distillation columns and batch processes*. Elsevier, pp. 327–332.
- Takeishi, Naoya and Alexandros Kalousis (2021). “Physics-integrated variational autoencoders for robust and interpretable generative modeling”. In *Advances in Neural Information Processing Systems* 34, pp. 14809–14821.
- Tathawadekar, Nilam N et al. (2023). “Incomplete to complete multiphysics forecasting: a hybrid approach for learning unknown phenomena”. In *Data-Centric Engineering* 4, e27.
- Thompson, Michael L and Mark A Kramer (1994). “Modeling chemical processes using prior knowledge and neural networks”. In *AIChE Journal* 40.8, pp. 1328–1340.
- Tulabandhula, Theja and Cynthia Rudin (2015). “Generalization bounds for learning with linear, polygonal, quadratic and conic side knowledge”. In *Machine Learning* 100.2, pp. 183–216.
- Vojří, Stanislav and Tomáš Kliegr (2020). “Editable machine learning models? A rule-based framework for user studies of explainability: S. Vojří, T. Kliegr”. In *Advances in Data Analysis and Classification* 14.4, pp. 785–799.
- Wehenkel, Antoine and Gilles Louppe (2019). “Unconstrained monotonic neural networks”. In *Advances in neural information processing systems* 32.
- Weinberger, Ethan, Joseph Janizek, and Su-In Lee (2020). “Learning deep attribution priors based on prior knowledge”. In *Advances in Neural Information Processing Systems* 33, pp. 14034–14045.
- Yang, Wanqian et al. (2019). “Output-constrained Bayesian neural networks”. In *arXiv preprint arXiv:1905.06287*.
- Yin, Yuan et al. (2021). “Augmenting physical models with deep networks for complex dynamics forecasting”. In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.
- Zaidan, Omar, Jason Eisner, and Christine Piatko (2007). “Using “annotator rationales” to improve machine learning for text categorization”. In *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference*, pp. 260–267.

- Zeng, Yang, Jin-Long Wu, and Heng Xiao (2021). “Enforcing imprecise constraints on generative adversarial networks for emulating physical systems”. In *Communications in Computational Physics* 30.3, pp. 635–665.
- Zhang, Haozhe, Dan Nettleton, and Zhengyuan Zhu (2019). “Regression-enhanced random forests”. In *arXiv preprint arXiv:1904.10416*.
- Zhang, Ye, Iain Marshall, and Byron C Wallace (2016). “Rationale-augmented convolutional neural networks for text classification”. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. Vol. 2016, p. 795.
- Zhang, Yifei et al. (2024). “MegI: Multimodal explanation-guided learning”. In *arXiv preprint arXiv:2411.13053*.
- Zhao, Runmin et al. (2024). “A two-stage CFD-GNN approach for efficient steady-state prediction of urban airflow and airborne contaminant dispersion”. In *Sustainable Cities and Society* 112, p. 105607.
- Zhong, Yaofeng Desmond, Biswadip Dey, and Amit Chakraborty (2019). “Symplectic ode-net: Learning hamiltonian dynamics with control”. In *arXiv preprint arXiv:1909.12077*.
- Zhou, Bolei et al. (2016). “Learning deep features for discriminative localization”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929.

3 HYBRID ADDITIVE MODELING FOR STATIC SUPERVISED REGRESSION

ADDENDUM

This chapter has previously been published as part of

Yann Claes, Vân Anh Huynh-Thu, and Pierre Geurts (2023). “Knowledge-guided additive modeling for supervised regression”. In *International Conference on Discovery Science*. Springer, pp. 64–78,

Yann Claes, Vân Anh Huynh-Thu, and Pierre Geurts (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.

Compared to the paper versions, for the version presented in this chapter, we restricted the contents to cover only the setting of *static* supervised regression problems. We moved the pseudo-codes for sequential and alternate optimization from the Appendix to Section 3.3.2, and added the results of gradient boosting and random forests for the experiment studying training sample size. Lastly, we aligned the notation with the other chapters.

In this chapter, we introduce a new training approach to learn hybrid additive models, based on partial dependence, an interpretation technique designed to understand the average behavior of a model for given features of interest. In comparison to recent hybrid additive training methods which introduce an intricate optimization process with sensitive regularization constraints, our method removes the need for such regularization through sequential optimization of the first-principles and data-driven models. We compare, on both synthetic and real regression problems, several approaches for training hybrid models against traditional machine learning methods. We investigate model-agnostic training procedures, hence experiments are carried out with different types of machine learning models, including tree-based models and neural networks. After the introduction of Section 3.1, we briefly review related works in Section 3.2. We then define the problem statement and the proposed methodology in Section 3.3, comparing the latter to existing hybrid training methods. In Section 3.4, we evaluate the performance of hybrid models obtained with our method, and we conclude on the observations in Section 3.5.

Contents

3.1	Introduction	26
3.2	Related work	26
3.3	Methodology	27
3.3.1	Problem statement	27

3.3.2	Training methods	28
3.4	Experiments	31
3.5	Conclusion	38
3.A	Model architectures	41

3.1 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 dataset. 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 with respect to domain knowledge (Daw et al., 2022; De Bézenac et al., 2019). 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 Von Rueden et al., 2021; Karniadakis et al., 2021; Willard et al., 2022 for reviews). A popular hybrid approach considers an additive combination of MB and ML components, and the success of these methods have been shown empirically on a range of synthetic and real-world problems (Yin, Le Guen, et al., 2021; Mehta et al., 2021; Donà et al., 2022). However, a significant challenge with such methods is to find the right balance between MB and ML components, if one is interested in retrieving a meaningful estimation of the parametric prior. This generally requires the introduction of regularization terms in the loss function, whose weighting coefficients are complex to optimize. Moreover, while these hybrid optimization methods have been mostly applied to dynamical systems, they have not been thoroughly studied in the context of standard regression problems, and the majority of ML models considered in these approaches are neural networks, leaving aside other methods. Our contributions are the following:

- We introduce a new hybrid training method based on partial dependence (PD), which avoids regularization constraints and makes fewer assumptions than other approaches.
- We investigate empirically the performance and benefits of hybrid methods against data-driven methods on *static* regression problems (in contrast 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 interaction between the MB and ML components. We compare different approaches for training such hybrid additive models, including our PD-based training. We highlight specific assumptions under which these approaches are expected to work well and relate their 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 hyper-parameters to tune, appear robust to the presence of irrelevant features and have been shown to often outperform neural networks on tabular data (Grinsztajn et al., 2022).

3.2 RELATED WORK

Hybrid additive modeling methods emerged several decades ago, combining first-principles models with different ML models. Already in the 1990’s, approaches derived by Kramer et al. (1992), Johansen et al. (1992), and Thompson et al. (1994) 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 (Yin, Ayed, et al., 2021; Qian et al., 2021; Takeishi et al., 2021; Wehenkel et al., 2022). In a more standard regression setting, J. Chen et al. (2007), Yu et al. (2010), and Zhang et al. (2019) combined a linear parametric term with a tree-based ML term.

On the challenge of finding the right balance between the MB and ML components, previous works have introduced regularization of the ML term to reduce parameter identification issues in the decomposition (Wouwer et al., 2004; Hu et al., 2011). Further works on this matter introduced constraints in the learning objective to better control contributions of the MB/ML components (Yin, Le Guen, et al., 2021; Donà et al., 2022). Elements of discussion about the well-posedness of this additive decomposition have also been introduced in previous works: Yin, Le Guen, et al. (2021) showed the existence and uniqueness of an optimal MB-ML pair when the contributions of the statistical model are constrained to be minimal, and Donà et al. (2022) demonstrated the convergence of an algorithm alternating between the optimization of both parts, without however any guarantee about convergence points. Nevertheless, most of these works only addressed dynamical systems forecasting.

3.3 METHODOLOGY

In this section, we formally define the problem of hybrid additive modeling in Section 3.3.1 and then describe various methods to address it in Section 3.3.2, including a new proposal based on partial dependence.

3.3.1 PROBLEM STATEMENT

Let us define a regression problem, where $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^d$, with $d \in \mathbb{N}_+$, are 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 parameter values 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 (Takeishi et al., 2021; Yin, Ayed, et al., 2021; Donà et al., 2022; Wehenkel et al., 2022).

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:

$$\mathcal{L}_{mse}(h, y) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} \{(h(\mathbf{x}) - y)^2\}, \quad (3.1)$$

$$\mathcal{L}_{phy}(h_k^{\theta_k}, f_k) = \mathbb{E}_{\mathbf{x}_k \sim p(\mathbf{x}_k)} \{(h_k^{\theta_k}(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\}. \quad (3.2)$$

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 $\mathcal{L}_{phy}$ measures how well the tuned h_k approximates f_k . The main

¹In 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.

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 $\mathcal{L}_{phy}$ is a loss that would compare the estimated and optimal parameters $\hat{\theta}_k$ and θ_k^* (e.g., $\|\hat{\theta}_k - \theta_k^*\|^2$). $\mathcal{L}_{phy}$ 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 $h_k^{\theta_k^*} = f_k$.

The following approximation of Equation (3.1) can be used as training objective:

$$\hat{\mathcal{L}}_{mse}(h, y; LS) = \frac{1}{N} \sum_{i=1}^N (h(\mathbf{x}_i) - y_i)^2. \quad (3.3)$$

Minimizing the distance in Equation (3.2) 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 Equation (3.1), Equation (3.2) 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 can make the problem easier. In the following, we will discuss the optimality of 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 $\mathbf{x}$.*

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

Problems where A2 is satisfied are easier as f_k captures all the dependence of y on $\mathbf{x}_k$. Furthermore, in problems where A3 is not met, it might be hard to distinguish real contributions from $\mathbf{x}_k$ to f from those due to correlations with features not in $\mathbf{x}_k$. Note that these assumptions do not strictly define our modeling framework, but are rather used in further developments and experiments as tools to derive claims about the optimality and convergence of hybrid training methods on problems that do (or do not) satisfy them.

3.3.2 TRAINING 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 each estimate 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.

Algorithm 1 Sequential Optimization: Static problems

Input: $LS = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$

$(\hat{\theta}_k, \hat{\gamma}) \leftarrow \text{fit}^{h_k+\gamma}(LS)$

$\hat{\theta}_a \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_i) - \hat{\gamma})\}_{i=1}^N)$

return $h_k^{\hat{\theta}_k} + \hat{\gamma} + h_a^{\hat{\theta}_a}$

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 by Zhang et al. (2019) (see Algorithm 1). More precisely, we first train $h_k^{\theta_k}$ on y by introducing a constant term $\gamma \in \mathbb{R}$, such that

$$(\hat{\theta}_k, \hat{\gamma}) = \text{fit}^{h_k+\gamma}(LS). \quad (3.4)$$

Afterwards, we fit h_a on the output residuals: $\hat{\theta}_a = \text{fit}^{h_a}\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k}(\mathbf{x}_i) - \hat{\gamma})\}_{i=1}^N$.

Introducing the constant term γ 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 $\hat{f}_k^* = \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathcal{L}_{mse}(\hat{f}_k, y)$ 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 $C = \mathbb{E}_{\mathbf{x}_a} \{f_a^r(\mathbf{x}_a)\}$. Indeed, as $y = f_k(\mathbf{x}_k) + f_a^r(\mathbf{x}_a) + \varepsilon$, we have:

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

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

$$\hat{f}_k^* = \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathbb{E}_{\mathbf{x}_k} \left\{ (\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2 \right\} - 2C \mathbb{E}_{\mathbf{x}_k} \left\{ (\hat{f}_k(\mathbf{x}_k) - f_k(\mathbf{x}_k)) \right\}, \quad (3.5)$$

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 Equation (3.5) 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 $\mathbf{x}_k \in \mathcal{X}_k$.

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 $f_a^r(\mathbf{x}_a) + \varepsilon$ acts as a pure additive noise term that needs to be averaged out during training. The approaches described in Sections 3.3.2 and 3.3.2 try to overcome this issue by fitting $h_k^{\theta_k}$ on corrected outputs that are expected to be closer to $f_k(\mathbf{x}_k)$. Regarding sequential optimization, without A2 and A3, the quality of the estimation $h_k^{\theta_k}$, according to Equation (3.2), is not guaranteed as there are problems satisfying A1 such that:

$$\exists \gamma \in \mathbb{R} : \arg \min_{\hat{f}_k \in \hat{\mathcal{F}}_k} \mathcal{L}_{mse}(\hat{f}_k, y) = f_k + \gamma. \quad (3.6)$$

An example of such problems will be given in Section 3.4.

ALTERNATE TRAINING OF h_k AND h_a

A hybrid additive approach was proposed by Donà et al. (2022), tackling the problem of dynamical systems forecasting, which alternates between updating h_k and updating h_a , using neural networks for h_a . They also introduce specific regularization criteria to try and find the optimal balance between both parts of the model, but we decided to focus on the non-regularized version of the algorithm, as we did not manage to significantly improve the results in the static experiments by adding the regularization process (such

process will however have an impact in the dynamical setting, see Chapter 4). Such alternate training was also proposed by J. Chen et al. (2007) and Yu et al. (2010) 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 (Breiman, 2001) and tree gradient boosting (Friedman, 2002).

The optimization process defined by Donà et al. (2022) is as follows: $\hat{\theta}_k$ is firstly initialized by (fully) fitting $h_k^{\hat{\theta}_k} + \gamma$ on y . Then, we alternate between: (1) a single epoch of gradient descent on $h_k^{\hat{\theta}_k} + \gamma$ and (2) either a single epoch for h_a (in the case of neural networks, as performed by Donà et al. (2022)) or a complete fit of h_a (in the case of tree-based models). This process is described in Algorithm 2.

Algorithm 2 Alternate Optimization: Static problems

Input: $LS = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $\eta > 0$

$\hat{\theta}_a^{(0)} \leftarrow$ Random Initialization
 $(\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \text{fit}^{h_k + \gamma}(LS)$
for $e = 1$ **to** N_e **do**
 $\hat{\theta}_a^{(e)} \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k^{(e-1)}}(\mathbf{x}_i) - \hat{\gamma}^{(e-1)})\}_{i=1}^N)$
 $(\hat{\theta}_k^{(e)}, \hat{\gamma}^{(e)}) \leftarrow \text{fit}^{h_k + \gamma}(\{(\mathbf{x}_i, y_i - h_a^{\hat{\theta}_a^{(e)}}(\mathbf{x}_i))\}_{i=1}^N)$
 $\hat{\theta}_a^{(N_e+1)} \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k^{(N_e)}}(\mathbf{x}_i) - \hat{\gamma}^{(N_e)})\}_{i=1}^N)$
return $h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)} + h_a^{\hat{\theta}_a^{(N_e+1)}}$

While some theoretical results are given by Donà et al. (2022), 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 version² 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.

PARTIAL DEPENDENCE-BASED TRAINING OF h_k AND h_a

We propose a novel approach relying on partial dependence (PD) functions (Friedman, 2001) 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:

$$PD(f, \mathbf{x}_k) = \mathbb{E}_{\mathbf{x}_{-k}} [f(\mathbf{x}_k, \mathbf{x}_{-k})] = \int f(\mathbf{x}_k, \mathbf{x}_{-k}) p(\mathbf{x}_{-k}) d\mathbf{x}_{-k}, \quad (3.7)$$

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 (Friedman, 2001):

$$PD(f, \mathbf{x}_k) = f_k(\mathbf{x}_k) + C, \text{ with } C = \mathbb{E}_{\mathbf{x}_a} \{f_a^r(\mathbf{x}_a)\}. \quad (3.8)$$

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 using Equation (3.7) requires in principle access to

²i.e., assuming an infinite training sample size and consistent estimators.

the input distribution, an approximation can be estimated from LS as follows:

$$\widehat{PD}(h_a, \mathbf{x}_k; LS) = \frac{1}{N} \sum_{i=1}^N h_a(\mathbf{x}_k, \mathbf{x}_{i,-k}), \quad (3.9)$$

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 $y - h_k^{\hat{\theta}_k}(\mathbf{x}_k) - \hat{y}$. We thus repeat the sequence of fitting h_a on the latter residuals, then fitting h_k on the obtained $\widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_k; LS) + h_k^{\hat{\theta}_k}(\mathbf{x}_k) + \hat{y}$, with $\hat{\theta}_k, \hat{y}$ and $\hat{\theta}_a$ the current optimized parameter vectors (see Algorithm 3).

Algorithm 3 Partial Dependence Optimization - Static problems

Input: $LS = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$
 $\hat{\theta}_a^{(0)} \leftarrow \text{fit}^{h_a}(LS)$
 $(\hat{\theta}_k^{(0)}, \hat{y}^{(0)}) \leftarrow \text{fit}^{h_k+\gamma}(\{(\mathbf{x}_{k,i}, \widehat{PD}(h_a^{\hat{\theta}_a^{(0)}}, \mathbf{x}_{k,i}; LS))\}_{i=1}^N)$
for $n = 1$ **to** N_r **do**
 $\hat{\theta}_a^{(n)} \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k^{(n-1)}}(\mathbf{x}_{k,i}) - \hat{y}^{(n-1)})\}_{i=1}^N)$
 $(\hat{\theta}_k^{(n)}, \hat{y}^{(n)}) \leftarrow \text{fit}^{h_k+\gamma}(\{(\mathbf{x}_{k,i}, h_k^{\hat{\theta}_k^{(n-1)}}(\mathbf{x}_{k,i}) + \hat{y}^{(n-1)} + \widehat{PD}(h_a^{\hat{\theta}_a^{(n)}}, \mathbf{x}_{k,i}; LS))\}_{i=1}^N)$
 $\hat{\theta}_a^{(N_r+1)} \leftarrow \text{fit}^{h_a}(\{(\mathbf{x}_i, y_i - h_k^{\hat{\theta}_k^{(N_r)}}(\mathbf{x}_{k,i}) - \hat{y}^{(N_r)})\}_{i=1}^N)$
return $h_k^{\hat{\theta}_k^{(N_r)}} + \hat{y}^{(N_r)} + h_a^{\hat{\theta}_a^{(N_r+1)}}$

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 Equation (3.8) 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.

3.4 EXPERIMENTS

We compare the different methods on several regression datasets, both simulated and real. Performance is measured through estimates of Equation (3.1) and Equation (3.2) (the latter only on simulated problems) on a test set TS , respectively denoted $\hat{\mathcal{L}}_{mse}(h, y; TS)$ and $\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$. 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 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 Section 3.3.2, under A2. 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 3.A. We monitor Equation (3.3) on a validation set and select the model that reaches the lowest validation loss.

INDEPENDENT INPUT FEATURES (A2 AND A3 SATISFIED)

Friedman problem We consider the following synthetic regression problem:

$$y = \theta_0 \sin(\theta_1 x_0 x_1) + \theta_2 (x_2 - \theta_3)^2 + \theta_4 x_3 + \theta_5 x_4 + \sum_{j=5}^9 0x_j + \varepsilon,$$

where $x_j \sim \mathcal{U}(0, 1)$, $j = 0, \dots, 9$, and $\varepsilon \sim \mathcal{N}(0, 1)$ (Friedman et al., 1983). We generate 10 different datasets using 10 different sets of values for $\theta_0, \dots, \theta_5$, 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. $f_k(\mathbf{x}_k) = \theta_0 \sin(\theta_1 x_0 x_1)$.

Table 3.1. Results on the *Friedman* problem. We report the mean and standard deviation of $\hat{\mathcal{L}}_{mse}$ and $\hat{\mathcal{L}}_{phy}$ 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	$\hat{\mathcal{L}}_{mse}(h, y; TS)$		$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$	
	Unfiltered	Filtered	Unfiltered	Filtered
MLP	$f_k \rightarrow h_a$	1.58 $\pm$ 0.33	1.23 $\pm$ 0.10	-
	Sequential	1.54 $\pm$ 0.31	1.43 $\pm$ 0.13	0.18 $\pm$ 0.16
	Alternate	1.43 $\pm$ 0.09	1.32 $\pm$ 0.09	0.10 $\pm$ 0.09 0.02 $\pm$ 0.02
	PD-based	1.54 $\pm$ 0.12	1.38 $\pm$ 0.09	0.06 $\pm$ 0.07
	h_a only	2.62 $\pm$ 0.75	-	-
GB	$f_k \rightarrow h_a$	1.73 $\pm$ 0.09	1.75 $\pm$ 0.12	-
	Sequential	1.74 $\pm$ 0.11	1.81 $\pm$ 0.14	0.18 $\pm$ 0.16
	Alternate	1.79 $\pm$ 0.11	1.78 $\pm$ 0.15	0.91 $\pm$ 1.45 0.06 $\pm$ 0.06
	PD-based	1.77 $\pm$ 0.13	1.78 $\pm$ 0.12	0.03 $\pm$ 0.02
	h_a only	3.43 $\pm$ 0.94	-	-
RF	$f_k \rightarrow h_a$	2.03 $\pm$ 0.18	1.96 $\pm$ 0.17	-
	Sequential	2.11 $\pm$ 0.23	2.05 $\pm$ 0.24	0.18 $\pm$ 0.16
	Alternate	2.03 $\pm$ 0.19	1.98 $\pm$ 0.17	0.04 $\pm$ 0.03 0.04 $\pm$ 0.04
	PD-based	2.16 $\pm$ 0.27	2.09 $\pm$ 0.26	0.16 $\pm$ 0.15
	h_a only	5.58 $\pm$ 1.91	-	-

We see in Table 3.1 that all hybrid training schemes outperform their data-driven counterpart. They come very close to the ideal $f_k \rightarrow h_a$ method, and are 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 Section 3.3.2). Filtering generally improves the performance of hybrid schemes as A2 is satisfied. PD-based optimization yields good approximations of f_k (as shown by a low $\hat{\mathcal{L}}_{phy}$). 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, as explained in Section 3.3.2. Filtering vastly decreases $\hat{\mathcal{L}}_{phy}$ for alternate approaches, supporting claims introduced in Section 3.3.2, while this measure remains unimpaired for sequential and PD-based training by construction.

CORRELATED INPUT FEATURES (A3 NOT SATISFIED)

Correlated Linear problem. 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 $\varepsilon \sim \mathcal{N}(0, 0.5^2)$. 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 (Greene, 2018):

$$\mathbb{E} \left[\hat{\beta}_0 \right] = \beta_0 + \frac{\text{cov}(x_0, x_1)}{\text{var}(x_0)} \beta_1. \quad (3.10)$$

We set $\text{cov}(x_0, x_1) = 2.25$ and $\text{var}(x_0) = 2$ so that Equation (3.10) reverses the sign of β_0 and Equation (3.6) is satisfied. The sequential approach should hence yield parameter estimates of β_0 close to Equation (3.10) while we expect the others to correct for this bias.

Table 3.2. Results for the *Correlated Linear* problem. We report $\hat{\mathcal{L}}_{mse}$ and $\text{rMAE}(\beta_0^*, \hat{\beta}_0)$, over 10 different datasets.

Method	$\hat{\mathcal{L}}_{mse}(h, y; TS)$		$\text{rMAE}(\beta_0^*, \hat{\beta}_0)$	
	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
	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
	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
	PD-based	0.41 ± 0.03	0.43 ± 0.04	57.47 ± 15.55

From Table 3.2, we observe that, contrary to PD-based training, the sequential and alternate methods return very bad estimations of β_0 , as A3 is no longer satisfied. Filtering corrects the bias for the alternate approach but degrades the MSE performance for sequential training as it removes the ability to compensate for the h_k misfit.

Table 3.3. Results for the *Correlated Friedman* problem.

Method	$\hat{\mathcal{L}}_{mse}(h, y; TS)$		$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$	
	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
	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
	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
	PD-based	3.24 ± 0.38	3.16 ± 0.37	0.27 ± 0.20
	h_a only	6.70 ± 1.47	-	-

Correlated Friedman problem. The structure is identical to the one in the previous Friedman problem, 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 the uncorrelated Friedman problem. Inputs are then scaled to be roughly in $[-1, 1]$. Here again, we use $f_k(\mathbf{x}_k) = \theta_0 \sin(\theta_1 x_0 x_1)$.

Table 3.3 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 yield poor h_k estimations without filtering since A3 is not met, but the latter mitigates this w.r.t. the former. Input filtering degrades predictive performance for the sequential methods as they cannot counterbalance a poor h_k .

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 $\hat{\beta} = \beta^*$, h_a still needs to compensate for $\sin(\gamma x_0)$. Therefore, filtering is expected to degrade performance for all hybrid approaches as $h_a(x_1)$ will never compensate for this gap, which we observe in Table 3.4, for both MLP and GB. Interestingly, sequential optimization seems to be the most impacted by filtering. It thus seems that having an interaction between the training of h_k and the training of h_a can still help to reduce the generalization error, even though $f_k(x_0)$ (and hence $h_k(x_0)$) is misspecified by design. Results for RF are not shown for the sake of space, but are similar to GB.

Table 3.4. Results for the *Overlapping* problem.

Method	$\hat{\mathcal{L}}_{mse}(h, y; TS)$		$\hat{\mathcal{L}}_{mse}(h, y; TS)$	
	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	

IMPACT OF TRAINING SAMPLE SIZE

In this experiment, we want to estimate the performance gap between hybrid methods and data-driven methods, for different sizes of the training sample. We measure the evolution of Equation (3.1) and Equation (3.2) for the different training algorithms, for increasing sample sizes of the *Correlated Friedman* problem. In particular, for a fixed training size, we generate 10 datasets using 10 different sets of values for $\theta_0, \dots, \theta_5$. The same 10 validation and test samples are fixed across all training sample sizes, containing 300 and 600 samples each.

Regarding MLPs, we observe in Figure 3.1 that the theoretically ideal $f_k \rightarrow h_a$ method generally performs better than any other training scheme, and significantly better for low-size regimes. Furthermore, hybrid models almost always outperform data-driven methods, except for PD-based training which struggles with very small sample sizes (30

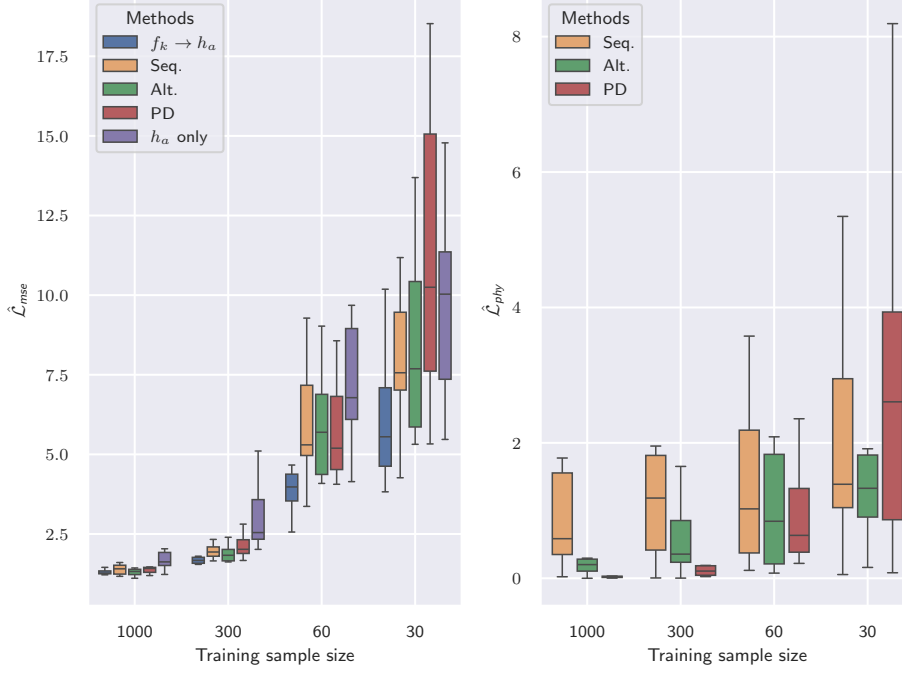


Figure 3.1. Evolution of $\hat{\mathcal{L}}_{mse}$ (left) and $\hat{\mathcal{L}}_{phy}$ (right) w.r.t. the training sample size, on the *Correlated Friedman* problem. The boxplot summarizes the results over 10 test sets, using MLP as h_a .

training samples). This could be explained by the fact that, in this algorithm, the training of h_k highly depends on the quality of fit of h_a , which serves as surrogate model for the true function f . Nevertheless, with slightly more training samples, we notice that it performs as good as the other training schemes. As could be expected, all training methods benefit from an increasing training sample size, and the gap between data-driven and hybrid models becomes smaller.

As observed for the generalization error, we also observe that the PD-based optimization yields worse estimates of h_k compared to the other hybrid approaches for very small sample sizes. However, it becomes competitive with the other methods at training samples of size 60, and eventually outperforms them at larger sample sizes.

Similarly to MLP, we observe in Figure 3.2 that hybrid methods with boosting outperform data-driven methods. Once again, PD-based training seems to struggle with very low sample sizes, although the gap with other hybrid schemes is more moderate and the performance of boosting alone is worse than that of MLP alone. Whatever the training sample size, the performance of hybrid methods with boosting is on par (and sometimes better) with that of MLP, both for generalization error and estimates of h_k .

The same observations can be made for RF in Figure 3.3, even though its performance is slightly worse than its MLP or boosting counterparts, whatever the training sample size. Still, it is interesting to note that hybrid methods using RF yield the best estimates of h_k for the smallest sample size.

3.4 EXPERIMENTS

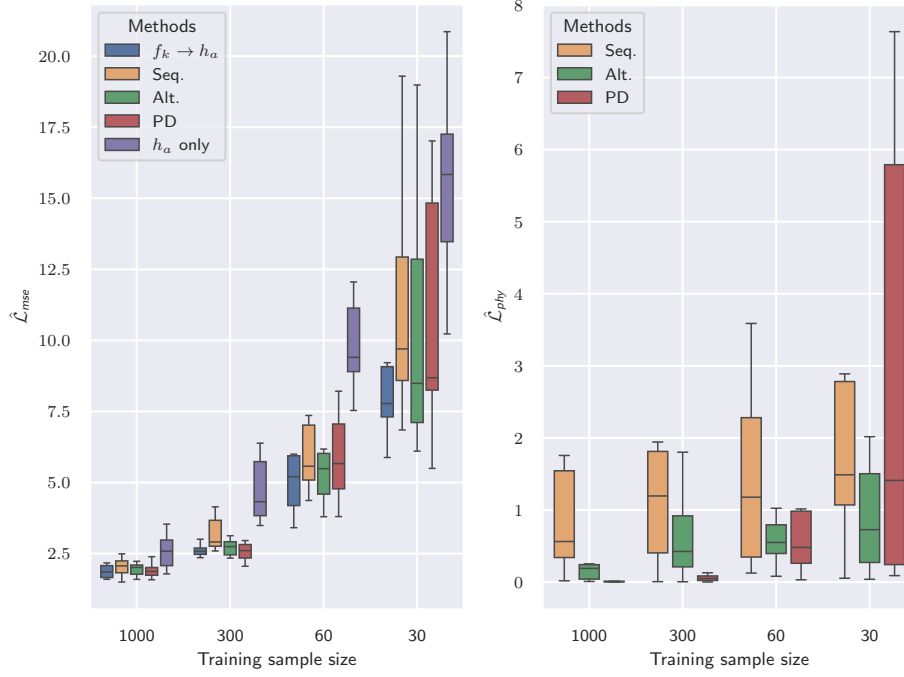


Figure 3.2. Evolution of $\hat{L}_{mse}$ (left) and $\hat{L}_{phy}$ (right) w.r.t. the training sample size, on the *Correlated Friedman* problem. The boxplot summarizes the results over 10 test sets, using GB as h_a .

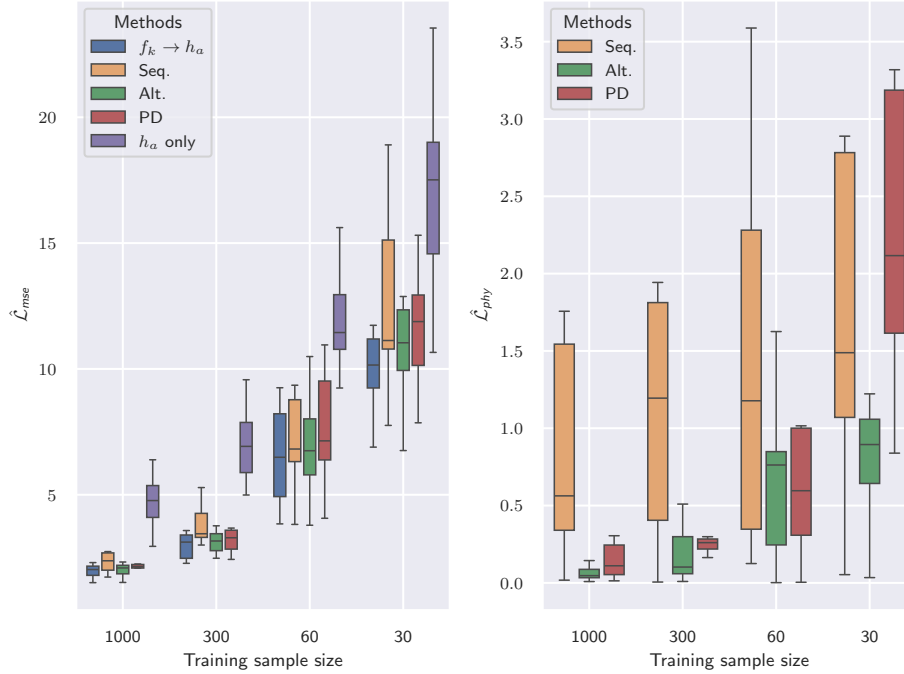


Figure 3.3. Evolution of $\hat{L}_{mse}$ (left) and $\hat{L}_{phy}$ (right) w.r.t. the training sample size, on the *Correlated Friedman* problem. The boxplot summarizes the results over 10 test sets, using RF as h_a .

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 $\mathcal{L}_{phy}$ in Equation (3.2) as it is deemed irrelevant.

We consider two settings for each dataset, inspired by Zhang et al. (2019). 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.

Table 3.5. 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 [-]

The features for both datasets are described in Table 3.5. The *Combined Cycle Power Plant* (CCPP) dataset (Tfekci et al., 2014) 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 (Yeh, 1998) is composed of 1,030 samples relating amounts of concrete components with the resulting compressive strength. As done in the work of Zhang et al. (2019), we introduce a new feature corresponding to the cement-to-water ratio.

From Table 3.6, 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 satisfied. 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

Table 3.6. Results for the real-world datasets. We report the mean and standard deviation of $\hat{\mathcal{L}}_{mse}$ 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

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.

3.5 CONCLUSION

We study several methods for training hybrid additive models on static supervised regression problems, for which 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 and verify their soundness on illustrative experiments. In particular, for such static additive problems where both terms are functions of disjoint and independent input features, we hypothesize that sequential and alternate training asymptotically converge to the true parametric component.

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 static 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, although the quality of its parametric estimator may be degraded when there is a clear overlap of input features between f_k and f_a . Another advantage of the PD-based training scheme is the reduced need for hyperparameter tuning compared to the alternate approach, which introduces multiple regularization coefficients that must be jointly optimized.

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 algebraic term, especially when filtering is relevant and can be applied.

REFERENCES

- Breiman, Leo (2001). “Random forests”. In *Machine learning* 45.1, pp. 5–32.
- Chen, Jinbo et al. (2007). “A partially linear tree-based regression model for assessing complex joint gene–gene and gene–environment effects”. In *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* 31.3, pp. 238–251.
- Chen, Tianqi and Carlos Guestrin (2016). “Xgboost: A scalable tree boosting system”. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.
- Claes, Yann, Vân Anh Huynh-Thu, and Pierre Geurts (2023). “Knowledge-guided additive modeling for supervised regression”. In *International Conference on Discovery Science*. Springer, pp. 64–78.
- (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.
- Daw, Arka et al. (2022). “Physics-guided neural networks (pgnn): An application in lake temperature modeling”. In *Knowledge guided machine learning*. Chapman and Hall/CRC, pp. 353–372.
- De Bézenac, Emmanuel, Arthur Pajot, and Patrick Gallinari (2019). “Deep learning for physical processes: Incorporating prior scientific knowledge”. In *Journal of Statistical Mechanics: Theory and Experiment* 2019.12, p. 124009.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Friedman, Jerome H (2001). “Greedy function approximation: a gradient boosting machine”. In *Annals of statistics*, pp. 1189–1232.
- (2002). “Stochastic gradient boosting”. In *Computational statistics & data analysis* 38.4, pp. 367–378.
- Friedman, Jerome H, Eric Grosse, and Werner Stuetzle (1983). “Multidimensional additive spline approximation”. In *SIAM Journal on Scientific and Statistical Computing* 4.2, pp. 291–301.
- Greene, W.H. (2018). “Econometric Analysis”. *Econometric Analysis*. Upper Saddle River, NJ: Pearson.
- Grinsztajn, Léo, Edouard Oyallon, and Gaël Varoquaux (2022). “Why do tree-based models still outperform deep learning on typical tabular data?” In *Advances in Neural Information Processing Systems* 35, pp. 507–520.
- Hu, Guanghao et al. (2011). “Hybrid modeling for the prediction of leaching rate in leaching process based on negative correlation learning bagging ensemble algorithm”. In *Computers & chemical engineering* 35.12, pp. 2611–2617.
- Johansen, Tor A and Bjarne A Foss (1992). “Representing and learning unmodeled dynamics with neural network memories”. In *IEEE American Control Conf.* Pp. 3037–3043.
- Karniadakis, George Em et al. (2021). “Physics-informed machine learning”. In *Nature Reviews Physics* 3.6, pp. 422–440.
- Kramer, Mark A, Michael L Thompson, and Phiroz M Bhagat (1992). “Embedding theoretical models in neural networks”. In *1992 American Control Conference*. IEEE, pp. 475–479.

- Mehta, Viraj et al. (2021). “Neural Dynamical Systems: Balancing Structure and Flexibility in Physical Prediction”. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 3735–3742.
- Paszke, Adam et al. (2019). “Pytorch: An imperative style, high-performance deep learning library”. In *Advances in neural information processing systems* 32.
- Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Qian, Zhaozhi et al. (2021). “Integrating expert ODEs into Neural ODEs: Pharmacology and disease progression”. In *Advances in Neural Information Processing Systems* 34, pp. 11364–11383.
- Takeishi, Naoya and Alexandros Kalousis (2021). “Physics-integrated variational autoencoders for robust and interpretable generative modeling”. In *Advances in Neural Information Processing Systems* 34, pp. 14809–14821.
- Tfekci, Pnar and Heysem Kaya (2014). “Combined Cycle Power Plant”. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5002N>.
- Thompson, Michael L and Mark A Kramer (1994). “Modeling chemical processes using prior knowledge and neural networks”. In *AIChE Journal* 40.8, pp. 1328–1340.
- Von Rueden, Laura et al. (2021). “Informed Machine Learning—A taxonomy and survey of integrating prior knowledge into learning systems”. In *IEEE Transactions on Knowledge and Data Engineering* 35.1, pp. 614–633.
- Wehenkel, Antoine et al. (2022). “Improving Generalization with Physical Equations”. In *Machine Learning and the Physical Sciences: Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS)*.
- Willard, Jared et al. (2022). “Integrating scientific knowledge with machine learning for engineering and environmental systems”. In *ACM Computing Surveys* 55.4, pp. 1–37.
- Wouwer, A Vande, Christine Renotte, and Ph Bogaerts (2004). “Biological reaction modeling using radial basis function networks”. In *Computers & chemical engineering* 28.11, pp. 2157–2164.
- Yeh, I-Cheng (1998). “Concrete Compressive Strength”. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5PK67>.
- Yin, Yuan, Ibrahim Ayed, et al. (2021). “LEADS: Learning dynamical systems that generalize across environments”. In *Advances in Neural Information Processing Systems* 34, pp. 7561–7573.
- Yin, Yuan, Vincent Le Guen, et al. (2021). “Augmenting physical models with deep networks for complex dynamics forecasting”. In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.
- Yu, Kai et al. (2010). “A partially linear tree-based regression model for multivariate outcomes”. In *Biometrics* 66.1, pp. 89–96.
- Zhang, Haozhe, Dan Nettleton, and Zhengyuan Zhu (2019). “Regression-enhanced random forests”. In *arXiv preprint arXiv:1904.10416*.

3.A MODEL ARCHITECTURES

Model hyperparameters for the static regression problems are reported in Table 3.7. These values have been chosen on the basis of preliminary small-scale experiments, with the main aim of obtaining a good tradeoff between predictive performance and computation times of the full experiments. Obviously better performance can be obtained for each of the methods with a more intensive hyper-parameter tuning. We used PyTorch (Paszke et al., 2019) for MLP, `scikit-learn` (Pedregosa et al., 2011) for RF and `xgboost` (T. Chen et al., 2016) 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 3.7. Model hyperparameters, for each experiment.

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

4 HYBRID ADDITIVE MODELING FOR DYNAMICAL SYSTEMS FORECASTING

ADDENDUM

This chapter has previously been published as part of

Yann Claes, Vân Anh Huynh-Thu, and Pierre Geurts (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.

Compared to the paper version, for the version presented in this chapter, we covered only the setting of dynamical systems forecasting. We clarified the pseudo-codes for joint and alternate optimization and moved them from the Appendix to Section 4.1.2. We also added some elements to motivate our PD-based optimization scheme for dynamical systems, and moved the experiments studying the impact of regularization and model initialization from the Appendix to Section 4.2.4. Lastly, we aligned the notation with the other chapters.

In the previous chapter, we introduced a new training approach to learn hybrid additive models, based on partial dependence (PD), and compared its performance against other hybrid training methods on static supervised regression problems. The present chapter extends our PD-based training approach to address the problem of dynamical systems forecasting and evaluates its performance against other hybrid additive training schemes. In Section 4.1, we start by formally defining the problem and explain how PD-based training can be extended in the context of dynamical systems forecasting. Then, Section 4.2 collects results on several dynamical systems, comparing our training approach to classical hybrid training methods. We also study the impact of model initialization and regularization for all methods. Finally, we conclude on these results in Section 4.3.

Contents

4.1	Methodology	44
4.1.1	Problem statement	44
4.1.2	Training methods	44
4.2	Experiments	47
4.2.1	Lotka-Volterra system	47
4.2.2	Damped Pendulum system	48
4.2.3	Reaction-Diffusion system	48
4.2.4	Impact of regularizer and model initialization	49
4.3	Conclusion	51
4.A	Model architectures	53

4.B	Dynamical experiments details	55
4.B.1	<i>Lotka-Volterra</i> system	55
4.B.2	<i>Damped Pendulum</i> system	55
4.B.3	<i>Reaction-Diffusion</i> system	56

4.1 METHODOLOGY

In this section, we formalize the considered setting of dynamical systems forecasting. Then, we present the hybrid training methods used as comparative baselines and extend the algorithm of Chapter 3 for the prediction of state trajectories.

4.1.1 PROBLEM STATEMENT

In this section, we target the problem of dynamical systems forecasting, i.e., predicting the evolution of systems whose dynamics are driven by:

$$\dot{\mathbf{x}}_t := \frac{d\mathbf{x}_t}{dt} = f(\mathbf{x}_t), \quad (4.1)$$

where the state $\mathbf{x}_t$ can be vector-valued, i.e., $\mathbf{x}_t \in \mathbb{R}^d$, or a d -dimensional vector field defined over some spatial domain, and $f(\mathbf{x}_t)$, the partially known dynamics function, has the same dimension as $\mathbf{x}_t$. We assume access to a finite set LS of N observed state trajectories over a given time interval $[0, T]$, with T the time horizon, i.e. $LS = \{\mathbf{x}_0^{(i)}, \dots, \mathbf{x}_T^{(i)}\}_{i=1}^N$. For clarity of notation, we use a single T although the time horizon can vary according to the considered system and between the N trajectories. Specifically, we consider systems whose dynamics f can be additively decomposed and we therefore extend Assumption A1 (defined in Chapter 3) as

Assumption 4 (A4, Additivity). $f(\mathbf{x}_t) = f_k(\mathbf{x}_{k,t}) + f_a(\mathbf{x}_t)$, where $\mathbf{x}_k$ is a subset of $K \leq d$ of the state variables.

As in Chapter 3, we assume partial knowledge of f through some parametric term $h_k^{\theta_k} \in \mathcal{H}_k$ with trainable parameters θ_k , such that for the optimal θ_k^* we have $h_k^{\theta_k^*} = f_k$. The residual function f_a is modeled by some ML component $h_a^{\theta_a} \in \mathcal{H}_a$, such that the final model $h \in \mathcal{H}$ is given by $h = h_k + h_a$, with $\mathcal{H} = \mathcal{H}_k + \mathcal{H}_a$. Similarly to the static setting, we introduce a constant term γ to the physical term $h_k^{\theta_k}$, such that we fit $h_k^{\theta_k} + \gamma$.

We seek to minimize the following two distances:

$$\mathcal{L}_{mse}(h, f) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \|h(\mathbf{x}) - f(\mathbf{x})\|^2, \quad (4.2)$$

$$\mathcal{L}_{phy}(h_k^{\theta_k}, f_k) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \{(h_k^{\theta_k}(\mathbf{x}_k) - f_k(\mathbf{x}_k))^2\}, \quad (4.3)$$

where $p(\mathbf{x})$ is the distribution over states. The first one measures the distance between the true dynamics f and its approximation h . The second distance $\mathcal{L}_{phy}$ measures how well the tuned h_k approximates f_k . In practice, minimizing Equation (4.2) translates into minimizing the distance between observed trajectories and trajectories obtained by integration of h :

$$\hat{\mathcal{L}}_{mse}(h, f; LS) = \frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T (\hat{\mathbf{x}}_t^{(i)} - \mathbf{x}_t^{(i)})^2, \quad (4.4)$$

where $\hat{\mathbf{x}}_t^{(i)}$ is the solution of the integration of h with initial condition $\mathbf{x}_0^{(i)}$ up to time t . In our experiments, we measure Equation (4.3) on some test set TS as:

$$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS) = \frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T (h_k^{\theta_k}(\mathbf{x}_{k,t}^{(i)}) - f_k(\mathbf{x}_{k,t}^{(i)}))^2. \quad (4.5)$$

4.1.2 TRAINING METHODS

Extending the algorithms of Chapter 3 to dynamical problems requires the integration of h to obtain the solutions $\hat{\mathbf{x}}_t^{(i)}$, which we do through the Neural ODE approach (Chen

et al., 2018). Contrary to the static experiments, we solely relied on ML models that allow backpropagation through time (allowing for easy integration with Neural ODE), thereby excluding tree-based models. For that reason, we can now introduce another hybrid training scheme in the comparisons, namely joint optimization of h_k and h_a , that was proposed by Yin et al. (2021). Note that this optimization algorithm could not be used in the static experiments due to the fact that tree-based methods cannot be trained jointly with the parametric component h_k .

Given the model approximation error observed on h_k for the sequential scheme in previous results (see Table 3.2 and Table 3.3 in Chapter 3), and assuming that most dynamical systems exhibit correlation between f_k and f_a through time, we choose to omit the sequential optimization scheme and find filtering irrelevant in such a configuration. Hence, we compare our PD-based hybrid scheme with the joint and alternate optimization of h_k and h_a , as defined by Yin et al. (2021) and Donà et al. (2022) respectively. As for the static case, we compare all schemes to the theoretically best setting of fitting h_a on $f - f_k$, i.e., assuming that the algebraic term f_k is perfectly known. We assume access to training functions $\text{fit}^{h_k+\gamma}$, $\text{fit}_{\text{NODE}}^{h_k+\gamma}$, $\text{fit}_{\text{NODE}}^{h_a}$ and $\text{fit}_{\text{NODE}}^{h_k+\gamma+h_a}$. The former estimates the parameters θ_k and γ by minimizing the MSE with the given target (e.g., the current PD estimation of h_a), whereas subsequent fit_{NODE} training functions represent a single step of gradient descent to minimize the MSE between the true trajectories and those integrated with Neural ODE.

JOINT OPTIMIZATION

In their work, Yin et al. (2021) optimize $h_k^{\theta_k}$ and $h_a^{\theta_a}$ jointly to minimize Equation (4.2). They introduce a regularizing criterion on h_a which penalizes the magnitude of its output, such that its contribution to the total model $h_k + h_a$ is kept minimal, ensuring existence and uniqueness of the minimizing pair (h_k, h_a) . Further details about the regularizer can be found in (Yin et al., 2021). We describe the non-regularized joint optimization process in Algorithm 4.

Algorithm 4 Joint Optimization: Non-Regularized Setting - Dynamical problems

Input: $LS = \{(\mathbf{x}_0^{(i)}, \dots, \mathbf{x}_T^{(i)})\}_{i=1}^N$

$\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}, \hat{\theta}_a^{(0)} \leftarrow$ Random Initialization

for $e = 1$ **to** N_e **do**

$(\hat{\theta}_k^{(e)}, \hat{\gamma}^{(e)}, \hat{\theta}_a^{(e)}) \leftarrow \text{fit}_{\text{NODE}}^{h_k+\gamma+h_a}(LS)$

return $h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)} + h_a^{\hat{\theta}_a^{(N_e)}}$

ALTERNATE OPTIMIZATION

As previously presented in Chapter 3, $(\hat{\theta}_k, \hat{\gamma})$ is initialized as the minimizer of $\mathcal{L}_{\text{mse}}(h_k^{\theta_k}, f)$, and this optimization scheme then alternates training steps between h_k and h_a . Donà et al. (2022) introduce an additional constraint on h_k to keep it close to its initialization in early training. The non-regularized training scheme (for dynamical systems) is described in Algorithm 5. A more complete description of the regularizer can be found in (Donà et al., 2022).

Algorithm 5 Alternate Optimization: Non-Regularized Setting - Dynamical problems

Input: $LS = \{(\mathbf{x}_0^{(i)}, \dots, \mathbf{x}_T^{(i)})\}_{i=1}^N$

$\hat{\theta}_a^{(0)} \leftarrow$ Random Initialization

$(\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \text{fit}_{\text{NODE}}^{h_k + \gamma}(LS)$

for $e = 1$ **to** N_e **do**

$\hat{\theta}_a^{(e)} \leftarrow \text{fit}_{\text{NODE}}^{h_a}(LS)$

$(\hat{\theta}_k^{(e)}, \hat{\gamma}^{(e)}) \leftarrow \text{fit}_{\text{NODE}}^{h_k + \gamma}(LS)$

$\hat{\theta}_a^{(N_e+1)} \leftarrow \text{fit}_{\text{NODE}}^{h_a}(LS)$

return $h_k^{\hat{\theta}_k^{(N_e)}} + \hat{\gamma}^{(N_e)} + h_a^{\hat{\theta}_a^{(N_e+1)}}$

PARTIAL-DEPENDENCE-BASED OPTIMIZATION

Our optimization scheme can be easily extended to the dynamical setting. Let us remind the definition of the PD of a function f on $\mathbf{x}_k$,

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

and, similarly to Assumption A2 (defined in Chapter 3), let us hypothesize that the dynamics f can be decomposed to satisfy

Assumption 5 (A5, Disjoint features). *Let $\mathbf{x}_{a,t}$ be a subset of state features disjoint from $\mathbf{x}_{k,t}$ ($\mathbf{x}_{k,t} \cap \mathbf{x}_{a,t} = \emptyset$) for a given timestep t . There exists a function $f_a^r(\mathbf{x}_{a,t})$ such that $f_a(\mathbf{x}_t) = f_a^r(\mathbf{x}_{a,t})$ for all $\mathbf{x}_t$.*

Then, we have that

$$PD(f, \mathbf{x}_{k,t}) = f_k(\mathbf{x}_{k,t}) + C, \text{ with } C = \mathbb{E}_{\mathbf{x}_{a,t}} \{f_a^r(\mathbf{x}_{a,t})\},$$

which motivates our approach in the context of dynamical systems forecasting.

In practice, we can compute an approximation from LS as follows:

$$\widehat{PD}(h_a, \mathbf{x}_{k,t}; LS) = \frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{\tau=1}^T h_a(\mathbf{x}_{k,t}, \mathbf{x}_{-k,\tau}^{(i)}), \quad (4.6)$$

where $\mathbf{x}_{-k,\tau}^{(i)}$ denotes the values of $\mathbf{x}_{-k}$ in the i -th trajectory of LS , at timestep τ .

We first fit any ML model $h_a^{\theta_a}$ on observed trajectories with Neural ODE, which we denote $\text{fit}_{\text{NODE}}^{h_a}$. At this stage, $h_a^{\theta_a}$ models the complete system dynamics f , hence it can be evaluated for multiple $\mathbf{x}_{k,t}$ values to obtain an initial approximation of $f_k(\mathbf{x}_{k,t})$ through $\widehat{PD}(h_a, \mathbf{x}_{k,t}; LS)$, on which we can fit $h_k^{\theta_k} + \gamma$ as for static regression problems, yielding an initial approximation $h_k^{\hat{\theta}_k} + \hat{\gamma}$. Due to potential residual dependencies, we then alternate between several epochs of fitting $h_a^{\theta_a}$ with Neural ODE to minimize $\mathcal{L}_{\text{mse}}(h_a^{\theta_a} + h_k^{\hat{\theta}_k} + \hat{\gamma}, f)$ (keeping $(\hat{\theta}_k, \hat{\gamma})$ fixed) and several epochs of fitting $h_k^{\theta_k} + \gamma$ on $\widehat{PD}(h_a^{\hat{\theta}_a}, \mathbf{x}_{k,t}; LS) + h_k^{\hat{\theta}_k} + \hat{\gamma}$, with $\hat{\theta}_k$, $\hat{\gamma}$ and $\hat{\theta}_a$ the current parameter vectors. The complete training procedure is summarized in Algorithm 6.

Algorithm 6 Partial Dependence Optimization - Dynamical problems

Input: $LS = \{(\mathbf{x}_0^{(i)}, \dots, \mathbf{x}_T^{(i)})\}_{i=1}^N$
 $\hat{\theta}_a^{(0)} \leftarrow \text{fit}_{\text{NODE}}^{h_a}(LS; h_k = 0)$
 $(\hat{\theta}_k^{(0)}, \hat{\gamma}^{(0)}) \leftarrow \text{fit}^{h_k+\gamma}(\{(\mathbf{x}_{k,t}^{(i)}, \widehat{PD}(h_a^{\hat{\theta}_a^{(0)}}, \mathbf{x}_{k,t}^{(i)}; LS))\}_{i,t=1}^{N,T})$
for $n = 1$ **to** N_r **do**
 $\hat{\theta}_a^{(n)} \leftarrow \text{fit}_{\text{NODE}}^{h_a}(LS; h_k = h_k^{\hat{\theta}_k^{(n-1)}} + \hat{\gamma}^{(n-1)})$
 $(\hat{\theta}_k^{(n)}, \hat{\gamma}^{(n)}) \leftarrow \text{fit}^{h_k+\gamma}(\{(\mathbf{x}_{k,t}^{(i)}, h_k^{\hat{\theta}_k^{(n-1)}}(\mathbf{x}_{k,t}^{(i)}) + \hat{\gamma}^{(n-1)} + \widehat{PD}(h_a^{\hat{\theta}_a^{(n)}}, \mathbf{x}_{k,t}^{(i)}; LS))\}_{i,t=1}^{N,T})$
 $\hat{\theta}_a^{(N_r+1)} \leftarrow \text{fit}_{\text{NODE}}^{h_a}(LS; h_k = h_k^{\hat{\theta}_k^{(N_r)}} + \hat{\gamma}^{(N_r)})$
return $h_k^{\hat{\theta}_k^{(N_r)}} + \hat{\gamma}^{(N_r)} + h_a^{\hat{\theta}_a^{(N_r+1)}}$

4.2 EXPERIMENTS

We compare the different training methods on three simulated dynamical systems that were used for evaluating related methods (Yin et al., 2021; Donà et al., 2022). We measure performance through log-estimates of Equation (4.4) and estimates of Equation (4.5) on a test set TS , which we respectively denote $\log\hat{\mathcal{L}}_{mse}(h, f; TS)$ and $\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$ (lower is better). For hybrid approaches, we use as h_a either a multilayer perceptron (MLP) or a convolutional neural network (CNN), depending on the system. We compare these hybrid models to a standard data-driven framework that uses only h_a . Architectures (e.g. for MLP, the number of layers and neurons) are optimized for each training method (unless clearly specified) on the validation set. Details about the hyperparameter optimization procedure and their optimal values are given in Appendix 4.A. We monitor Equation (4.4) on a validation set and select the model that reaches the lowest validation loss. Datasets and optimization details are provided in Appendix 4.B. For conciseness, we only report results obtained using the optimization scheme as exactly described in each reference work and compare the impact of using the specific regularization criterion or not (if relevant).

4.2.1 LOTKA-VOLTERRA SYSTEM

Lotka-Volterra equations model the evolution of preys x and predators y according to:

$$\begin{cases} \dot{x} &= \alpha x - \beta xy \\ \dot{y} &= -\delta y + \gamma xy \end{cases} \iff \begin{cases} \dot{p} &= \alpha - \beta e^q \\ \dot{q} &= -\delta + \gamma e^p \end{cases}$$

where $p = \ln(x)$ and $q = \ln(y)$. Due to training stability reasons, we did not manage to fit any method on the left-hand side system. Therefore, we choose to focus on the right-hand side system, where $\mathbf{x} = (p, q)$ is the system state. We assume $h_k^{\theta_k}(p, q) = (-\theta_k^1 e^q, 0)$ and $h_a^{\theta_a}$ accounts for $(\alpha, -\delta + \gamma e^p)$.

In terms of trajectory estimation, we can observe from Table 4.1 that all hybrid optimization schemes perform very similarly, although alternate training performs slightly better. We can also note that all training methods yield better estimators than the purely data-driven setting. As concerns the fitting of h_k , results suggest that our PD-based training procedure produces a more robust and accurate estimation of f_k .

We can also study the effect of the regularizers introduced by Yin et al. (2021) and Donà et al. (2022). For this specific problem, joint and alternate optimization combined with

Table 4.1. Results for the *Lotka-Volterra* system, averaged over 5 different initializations of h . Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Init. (resp. No Init.) refers to initializing h_k as the minimizer of $\mathcal{L}_{mse}(h_k, f)$ (resp. randomly).

Method	$\log-\hat{\mathcal{L}}_{mse}(h, f; TS)$		$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$	
	No Reg.	Reg.	No Reg.	Reg.
$f_k \rightarrow h_a$	-5.45 ± 0.02		-	
Joint (No Init.)	-5.22 ± 0.04	-5.45 ± 0.06	$2.51 \times 10^{-1} \pm 3.90 \times 10^{-2}$	$1.17 \times 10^{-6} \pm 1.47 \times 10^{-13}$
Alternate (Init.)	-5.19 ± 0.04	-5.52 ± 0.05	$3.79 \times 10^{-1} \pm 4.49 \times 10^{-2}$	$1.35 \times 10^{-1} \pm 7.19 \times 10^{-3}$
PD-based (No Init.)	-5.41 ± 0.04		$1.05 \times 10^{-7} \pm 9.51 \times 10^{-15}$	
h_a only	-5.02 ± 0.09		-	

their regularizers seem to improve both the estimation of f_k and the final predictive performance.

4.2.2 DAMPED PENDULUM SYSTEM

The evolution of the angle $\theta(t)$ of a damped pendulum system is described as follows:

$$\frac{d^2\theta}{dt^2} + \omega_0^2 \sin \theta + \xi \frac{d\theta}{dt} = 0, \quad \Longleftrightarrow \quad \begin{cases} \dot{\theta} &= \frac{d\theta}{dt} \\ \ddot{\theta} &= -\omega_0^2 \sin \theta - \xi \frac{d\theta}{dt} \end{cases}$$

where $\mathbf{x} = (\theta, \frac{d\theta}{dt})$ is the system state. We assume $h_k^{\theta_k}(\theta, \frac{d\theta}{dt}) = (\frac{d\theta}{dt}, -\theta_k \sin \theta)$, and $h_a^{\theta_a}$ accounts for the damping term $-\xi d\theta/dt$.

Table 4.2. Results for the *Damped Pendulum* system, averaged over 5 different initializations of h . Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Init. (resp. No Init.) refers to initializing h_k as the minimizer of $\mathcal{L}_{mse}(h_k, f)$ (resp. randomly).

Method	$\log-\hat{\mathcal{L}}_{mse}(h, f; TS)$		$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$	
	No Reg.	Reg.	No Reg.	Reg.
$f_k \rightarrow h_a$	-8.75 ± 0.01		-	
Joint (No Init.)	-7.68 ± 0.03	-5.98 ± 0.04	$2.29 \times 10^{-1} \pm 4.47 \times 10^{-3}$	$5.51 \times 10^{-3} \pm 2.36 \times 10^{-6}$
Alternate (Init.)	-7.36 ± 0.02	-6.91 ± 0.00	$1.61 \times 10^{-1} \pm 4.09 \times 10^{-3}$	$5.90 \times 10^{-3} \pm 2.64 \times 10^{-6}$
PD-based (No Init.)	-8.28 ± 0.04		$2.42 \times 10^{-5} \pm 4.57 \times 10^{-12}$	
h_a only	-7.28 ± 0.01		-	

Table 4.2 highlights that our PD-based optimization predicts more accurate trajectories than the other hybrid training methods. Similarly, all training methods yield better estimators than the purely data-driven setting, provided we do not regularize. For this system, we see that the baseline $f_k \rightarrow h_a$ is now the optimal training approach. Concerning the estimation of h_k , we observe that our hybrid training scheme yields the best estimations. Contrary to the previous system, regularizing joint and alternate optimization improves the resulting h_k , but now degrades final predictive performance. This underlines the fact that there exists an interesting trade-off to investigate for both methods, but also that they are very sensitive to the chosen regularization hyperparameter values.

4.2.3 REACTION-DIFFUSION SYSTEM

This 2-dimensional system is driven by the following PDEs (Klaasen et al., 1984):

$$\begin{cases} \frac{\delta u}{\delta t} &= \alpha \Delta u + R_u(u, v; \gamma) \\ \frac{\delta v}{\delta t} &= \beta \Delta v + R_v(u, v) \end{cases}$$

where $\mathbf{x} = (u, v)$ is the system state defined over some rectangular domain with circular boundary conditions. Δ is the Laplace operator and α, β are the diffusion coefficients for both quantities. $R_u(u, v; \gamma) = u - u^3 - \gamma - v$ and $R_v(u, v) = u - v$ are the local reaction terms. We assume $h_k^{\theta_k}(u, v) = (\theta_k^1 \Delta u, \theta_k^2 \Delta v)$, and $h_a^{\theta_a}$ accounts for the reaction terms R_u and R_v . Contrary to the two previous experiments, due to time and computational constraints, we did not optimize hyperparameters. Instead, all training schemes were given the same set of hyperparameters that we deemed reasonable.

Table 4.3. Results for the *Reaction-Diffusion* system, averaged over 5 different initializations of h . Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Init. (resp. No Init.) refers to initializing h_k as the minimizer of $\mathcal{L}_{mse}(h_k, f)$ (resp. randomly).

Method	$\log-\hat{\mathcal{L}}_{mse}(h, f; TS)$		$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$	
	No Reg.	Reg.	No Reg.	Reg.
$f_k \rightarrow h_a$	-3.43 ± 0.11		-	
Joint (No Init.)	-3.62 ± 0.02	-2.96 ± 0.01	$1.17 \times 10^{-5} \pm 1.42 \times 10^{-10}$	$5.45 \times 10^{-4} \pm 2.64 \times 10^{-7}$
Alternate (Init.)	-2.99 ± 0.02	-2.97 ± 0.01	$4.80 \times 10^{-5} \pm 2.81 \times 10^{-9}$	$1.89 \times 10^{-3} \pm 1.67 \times 10^{-6}$
PD-based (No Init.)	-3.09 ± 0.02		$1.64 \times 10^{-3} \pm 1.01 \times 10^{-7}$	
h_a only	-2.86 ± 0.01		-	

From Table 4.3, we observe that the best performing scheme is the non-regularized joint training of h_k and h_a , as opposed to the previous problems. As before, introducing the regularizers defined in both reference works induces a drop in final performance, still more controlled for alternate optimization. This drop is also observed in the estimation of f_k (which seems optimal for the non-regularized configuration), confirming that such regularizing criteria are very sensitive to optimize.

PD-based training yields reasonable trajectory estimates, but the quality of its estimation of h_k is no longer the best amongst all. Indeed, contrary to the two previous systems, it is clear that A5 is no longer satisfied as $f_a(u, v) = (u - u^3 - \gamma - v, u - v)$ includes variables present in f_k . The latter setting represents a limitation of the PD-based optimization scheme as its estimation of the physical component is now effectively impacted, compared to the other hybrid training methods.

4.2.4 IMPACT OF REGULARIZER AND MODEL INITIALIZATION

In this section, we report results for variants of the optimization methods presented in Section 4.1.2. In particular, we test the initialization introduced by Donà et al. (2022) with both joint and PD-based training. For PD-based training, this means replacing the first two steps of Algorithm 6 with a fit of $(\hat{\theta}_k, \hat{\gamma})$ to minimize $\mathcal{L}_{mse}(h_k^{\theta_k} + \gamma, f)$. We also compare the impact of regularizing h_a for both joint and alternate optimizations.

LOTKA-VOLTERRA SYSTEM

Compared to Section 4.2.1, Table 4.4 informs us that, for this problem, PD-based training can yield models equally good to alternate optimization, provided we initialize h_k as specifically defined by Donà et al. (2022). We can also note that the resulting estimation of h_k is even better. Moreover, joint training with no regularization reaches similar predictive performance, at the cost of a worse estimation of h_k (although better than that of alternate training). Given that the parametric component we fit is $h_k^{\theta_k} + \gamma$ (with $\gamma \in \mathbb{R}$), initializing h_k as $(\hat{\theta}_k, \hat{\gamma}) = \arg \min_{\theta_k, \gamma} \mathcal{L}_{mse}(h_k^{\theta_k} + \gamma, f)$ actually makes γ converge to α , which explains why training methods seem to benefit from this initialization in this particular system.

We also investigated the impact of removing this specific h_k initialization from the alternate scheme. For this problem, it appeared clearly that such initialization enhanced the final model h .

Table 4.4. Results for other *Lotka-Volterra* system, averaged over 5 different initializations of h . Init. refers to $(\hat{\theta}_k, \hat{\gamma}) = \arg \min_{\theta_k, \gamma} \mathcal{L}_{mse}(h_k^{\theta_k} + \gamma, f)$ while Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Methods not included in the table in the main paper are in italics.

Method	$\log-\hat{\mathcal{L}}_{mse}(h, f; TS)$	$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$
$f_k \rightarrow h_a$	-5.45 ± 0.02	-
Joint - No Init. & No Reg.	-5.22 ± 0.04	$2.51 \times 10^{-1} \pm 3.90 \times 10^{-2}$
Joint - No Init. & Reg.	-5.45 ± 0.06	$1.17 \times 10^{-6} \pm 1.47 \times 10^{-13}$
<i>Joint - Init. & No Reg.</i>	-5.53 ± 0.01	$8.62 \times 10^{-2} \pm 2.63 \times 10^{-3}$
<i>Joint - Init. & Reg.</i>	-5.42 ± 0.04	$1.66 \times 10^{-6} \pm 7.63 \times 10^{-13}$
<i>Alternate - No Init. & No Reg.</i>	-5.23 ± 0.09	$5.58 \times 10^{-1} \pm 1.64 \times 10^{-1}$
<i>Alternate - No Init. & Reg.</i>	-5.32 ± 0.01	$1.66 \times 10^{-1} \pm 5.00 \times 10^{-4}$
<i>Alternate - Init. & No Reg.</i>	-5.19 ± 0.04	$3.79 \times 10^{-1} \pm 4.49 \times 10^{-2}$
<i>Alternate - Init. & Reg.</i>	-5.52 ± 0.05	$1.35 \times 10^{-1} \pm 7.19 \times 10^{-3}$
PD-based - No Init.	-5.41 ± 0.04	$1.05 \times 10^{-7} \pm 9.51 \times 10^{-15}$
<i>PD-based - Init.</i>	-5.51 ± 0.02	$6.27 \times 10^{-8} \pm 8.72 \times 10^{-15}$
h_a only	-5.02 ± 0.09	-

Table 4.5. Results for the *Damped Pendulum* system, averaged over 5 different initializations of h . Init. refers to $(\hat{\theta}_k, \hat{\gamma}) = \arg \min_{\theta_k, \gamma} \mathcal{L}_{mse}(h_k^{\theta_k} + \gamma, f)$ while Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Methods not included in the table in the main paper are in italics.

Method	$\log-\hat{\mathcal{L}}_{mse}(h, f; TS)$	$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$
$f_k \rightarrow h_a$	-8.75 ± 0.01	-
Joint - No Init. & No Reg.	-7.68 ± 0.03	$2.29 \times 10^{-1} \pm 4.47 \times 10^{-3}$
Joint - No Init. & Reg.	-5.98 ± 0.04	$5.51 \times 10^{-3} \pm 2.36 \times 10^{-6}$
<i>Joint - Init. & No Reg.</i>	-7.74 ± 0.02	$1.57 \times 10^{-1} \pm 3.75 \times 10^{-3}$
<i>Joint - Init. & Reg.</i>	-5.83 ± 0.03	$2.83 \times 10^{-2} \pm 2.26 \times 10^{-3}$
<i>Alternate - No Init. & No Reg.</i>	-7.23 ± 0.04	$2.34 \times 10^{-1} \pm 4.47 \times 10^{-3}$
<i>Alternate - No Init. & Reg.</i>	-6.91 ± 0.00	$6.84 \times 10^{-3} \pm 4.56 \times 10^{-6}$
<i>Alternate - Init. & No Reg.</i>	-7.36 ± 0.02	$1.61 \times 10^{-1} \pm 4.09 \times 10^{-3}$
<i>Alternate - Init. & Reg.</i>	-6.90 ± 0.00	$5.90 \times 10^{-3} \pm 2.64 \times 10^{-6}$
PD-based - No Init.	-8.28 ± 0.04	$2.42 \times 10^{-5} \pm 4.57 \times 10^{-12}$
<i>PD-based - Init.</i>	-8.29 ± 0.02	$2.36 \times 10^{-5} \pm 5.56 \times 10^{-12}$
h_a only	-7.28 ± 0.01	-

DAMPED PENDULUM SYSTEM

Similarly to Section 4.2.2, we can see from Table 4.5 that PD-based training reaches the highest performance both for trajectory estimation as well as for the estimation of h_k . For this system, the particular h_k initialization does not seem to impact our training method, on any level.

There is now more nuance to the impact of such initialization on joint optimization. Indeed, while it seems to help the non-regularized configuration, the effect is the opposite when we add the presence of their regularizer, indicating that hyperparameters could have been optimized even more or simply that their regularization criterion is not suited with this specific initialization. However, we can observe that the effect is positive for models optimized in an alternate fashion.

Table 4.6. Results for the *Reaction-Diffusion* system, averaged over 5 different initializations of h . Init. refers to $(\hat{\theta}_k, \hat{\gamma}) = \arg \min_{\theta_k, \gamma} \mathcal{L}_{mse}(h_k^{\theta_k} + \gamma, f)$ while Reg. (resp. No Reg.) refers to using (resp. not using) the specific regularizer. Methods not included in the table in the main paper are in *italics*.

Method	$\log-\hat{\mathcal{L}}_{mse}(h, f; TS)$	$\hat{\mathcal{L}}_{phy}(h_k^{\theta_k}, f_k; TS)$
$f_k \rightarrow h_a$	-3.43 ± 0.11	-
Joint - No Init. & No Reg.	-3.62 ± 0.02	$1.17 \times 10^{-5} \pm 1.42 \times 10^{-10}$
Joint - No Init. & Reg.	-2.96 ± 0.01	$5.45 \times 10^{-4} \pm 2.64 \times 10^{-7}$
<i>Joint - Init. & No Reg.</i>	-3.14 ± 0.11	$4.43 \times 10^{-5} \pm 2.61 \times 10^{-9}$
<i>Joint - Init. & Reg.</i>	-2.65 ± 0.04	$1.14 \times 10^{-3} \pm 2.50 \times 10^{-7}$
<i>Alternate - No Init. & No Reg.</i>	-3.26 ± 0.02	$1.94 \times 10^{-4} \pm 6.95 \times 10^{-9}$
<i>Alternate - No Init. & Reg.</i>	-2.98 ± 0.02	$1.71 \times 10^{-4} \pm 6.18 \times 10^{-9}$
<i>Alternate - Init. & No Reg.</i>	-2.99 ± 0.02	$4.80 \times 10^{-5} \pm 2.81 \times 10^{-9}$
<i>Alternate - Init. & Reg.</i>	-2.97 ± 0.01	$1.89 \times 10^{-3} \pm 1.67 \times 10^{-6}$
PD-based - No Init.	-3.09 ± 0.02	$1.64 \times 10^{-3} \pm 1.01 \times 10^{-7}$
<i>PD-based - Init.</i>	-3.12 ± 0.02	$2.66 \times 10^{-3} \pm 2.51 \times 10^{-6}$
h_a only	-2.86 ± 0.01	-

REACTION-DIFFUSION SYSTEM

For this 2-D problem, Table 4.6 suggests that non-regularized joint optimization with uniform h_k initialization performs best, as could be seen from Table 4.3. As was said, PD-based training seems to struggle now that there is a clear overlap between f_k and f_a . Furthermore, the particular h_k initialization does not seem to impact our training method.

As concerns joint optimization, the initialization proposed by Donà et al. (2022) clearly degrades both performance criteria. Nevertheless, we can observe that once again the effect is positive for regularized alternate training, indicating that, as could be observed from previous systems, there is a very sensitive balance to find between the initialization of the physical component and the regularizers, along with their optimal coefficient values.

4.3 CONCLUSION

We extend our partial-dependence-based hybrid training approach to the problem of dynamical systems forecasting and show that PD-based training also yields robust and accurate models in such setting. Once again, we emphasize its positive effects in estimating faithful parametric priors, yielding competitive hybrid models while removing the need for intricate regularization and reducing the load of hyperparameter tuning.

More generally, we can highlight from our experiments that hybrid models usually outperform data-driven models. Such hybrid methods yield more interpretable and accurate models than purely data-driven models.

As future work, we are interested in studying theoretical conditions of optimality for our PD-based hybrid training and other approaches, in the context of dynamical systems. We also plan to try and extend current hybrid methods tackling dynamical systems forecasting so that they can be used with tree-based methods.

REFERENCES

- Chen, Ricky T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In *Advances in Neural Information Processing Systems*.
- Claes, Yann, Vân Anh Huynh-Thu, and Pierre Geurts (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In *arXiv preprint arXiv:1412.6980*.
- Klaasen, Gene A and William C Troy (1984). “Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations”. In *SIAM Journal on Applied Mathematics* 44.1, pp. 96–110.
- Yin, Yuan et al. (2021). “Augmenting physical models with deep networks for complex dynamics forecasting”. In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.

4.A MODEL ARCHITECTURES

Model hyperparameters for the dynamical forecasting problems are reported in Tables 4.8, 4.9 and 4.10. We used PyTorch for MLP and CNN. Tested hyperparameters are reported in Table 4.7. Given that we only optimized hyperparameters for the *Lotka-Volterra* and *Damped Pendulum* systems, architectural hyperparameters solely refer to those of MLPs. According to the algorithm described in (Yin et al., 2021) and (Donà et al., 2022), λ_h is optimized for joint optimization whereas λ_h , λ_{h_k} and λ_{h_a} are optimized for alternate training.

Table 4.7. Hyperparameter values tested for the *Lotka-Volterra* and *Damped Pendulum* systems.

Hyperparameter			
Learning Rate (LR)	5×10^{-4}	1×10^{-4}	1×10^{-5}
Num. hidden layers	1	2	-
Hidden size	16	32	64
λ_h - Joint	1	10	20
λ_h - Alternate	1×10^{-1}	1×10^{-2}	1×10^{-3}
λ_{h_k} - Alternate	1×10^{-1}	1×10^{-2}	1×10^{-3}
λ_{h_a} - Alternate	1×10^{-1}	1×10^{-2}	1×10^{-3}

For each problem, for a given optimization scheme and for each of the five different initializations of h , we tested every combination of hyperparameters and evaluated it on the corresponding validation set. As optimal hyperparameters may vary from one seed to the other, we checked all five optimal configurations of the given training method and reported the most frequent one. When no clear majority could be established, we reported several optimal values for the same hyperparameter. Note that the optimal hyperparameters across the five seeds were most of the time identical.

Table 4.8. Optimal model hyperparameters (MLP), for the *Lotka-Volterra* system.

Method	LR	Num. layers	Hidden size	λ_h	λ_{h_k}	λ_{h_a}
$f_k \rightarrow h_a$	5×10^{-4}	2	64	-	-	-
Joint - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - No Init. & Reg.	1×10^{-5}	2	64	10/20	-	-
Joint - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - Init. & Reg.	1×10^{-5}	2	64	20	-	-
Alternate - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - No Init. & Reg.	5×10^{-4}	2	64	0.1	0.001	0.001
Alternate - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - Init. & Reg.	5×10^{-4}	2	64	0.1	0.01	0.001
PD-based - No Init.	5×10^{-4}	2	64	-	-	-
PD-based - Init.	5×10^{-4}	2	64	-	-	-
h_a only	5×10^{-4}	2	64	-	-	-

4.A MODEL ARCHITECTURES

Table 4.9. Optimal model hyperparameters (MLP), for the *Damped Pendulum* system.

Method	LR	Num. layers	Hidden size	λ_h	λ_{h_k}	λ_{h_a}
$f_k \rightarrow h_a$	5×10^{-4}	2	64	-	-	-
Joint - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - No Init. & Reg.	1×10^{-4}	1/2	32/16	20	-	-
Joint - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Joint - Init. & Reg.	1×10^{-4}	1/2	16	20	-	-
Alternate - No Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - No Init. & Reg.	5×10^{-4}	2	64	0.1	0.001	0.001
Alternate - Init. & No Reg.	5×10^{-4}	2	64	-	-	-
Alternate - Init. & Reg.	5×10^{-4}	2	64	0.1	0.001	0.001
PD-based - No Init.	5×10^{-4}	2	64	-	-	-
PD-based - Init.	5×10^{-4}	2	64	-	-	-
h_a only	5×10^{-4}	2	64	-	-	-

Table 4.10. Model hyperparameters (CNN), for the *Reaction-Diffusion* system.

Method	LR	Num. layers	Hidden channels	λ_h	λ_{h_k}	λ_{h_a}
$f_k \rightarrow h_a$	1×10^{-4}	3	8	-	-	-
Joint - No Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Joint - No Init. & Reg.	1×10^{-4}	3	8	20	-	-
Joint - Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Joint - Init. & Reg.	1×10^{-4}	3	8	20	-	-
Alternate - No Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Alternate - No Init. & Reg.	1×10^{-4}	3	8	0.1	0.01	0.001
Alternate - Init. & No Reg.	1×10^{-4}	3	8	-	-	-
Alternate - Init. & Reg.	1×10^{-4}	3	8	0.1	0.01	0.001
PD-based - No Init.	1×10^{-4}	3	8	-	-	-
PD-based - Init.	1×10^{-4}	3	8	-	-	-
h_a only	1×10^{-4}	3	8	-	-	-

4.B DYNAMICAL EXPERIMENTS DETAILS

4.B.1 LOTKA-VOLTERRA SYSTEM

The system is driven by the following dynamics:

$$\begin{cases} \dot{x} &= \alpha x - \beta xy \\ \dot{y} &= -\delta y + \gamma xy \end{cases}$$

where prey and predator quantities x and y can be transformed as $p = \ln(x)$, $q = \ln(y)$ such that

$$\begin{cases} \dot{p} &= \alpha - \beta e^q \\ \dot{q} &= -\delta + \gamma e^p \end{cases}$$

where $\mathbf{x} = (p, q)$ is the system state.

We simulate a dataset of 200 trajectories spanning 401 timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 20]$ s. We first generate initial prey-predator proportions $(x, y) \sim \mathcal{U}[0, 1]^2$ and compute the resulting initial states $\mathbf{x}_0 = (\ln(x), \ln(y))$. We set $\alpha = \beta = \gamma = \delta = 1$ and generate trajectories by integrating for 400 steps ($\delta t = 0.05$) using a 4th order Runge-Kutta scheme, with an integration timestep $\delta t_{sim} = 0.001$. We split the dataset into train-validation-test subsets (50/25/25%). The train set is eventually obtained by extracting windows of 41 timesteps (with a stride of 2) from the original train split, yielding 18 100 smaller sub-trajectories.

For the data-driven, joint and alternate schemes, models are trained for 500 epochs using the Euler integrator with $\delta t = 0.05$. For the PD-based optimization scheme, each training consists of 50 epochs repeated 10 times, with the same integration settings. The final training of h_a consists of 150 epochs. All methods are optimized with Adam (Kingma et al., 2014).

4.B.2 DAMPED PENDULUM SYSTEM

The damped pendulum dynamics are described by:

$$\begin{cases} \dot{\theta} &= \frac{d\theta}{dt} \\ \ddot{\theta} &= -\omega_0^2 \sin \theta - \xi \frac{d\theta}{dt} \end{cases}$$

where $\mathbf{x} = (\theta, \frac{d\theta}{dt})$ is the system state.

We simulate a dataset of 200 trajectories spanning 201 timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 10]$ s. We first generate the initial states $\mathbf{x}_0 = (\theta_0, \frac{d\theta_0}{dt}) \sim \mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}] \times \mathcal{U}[0, 0.1]$. We set $(\omega, \xi) \sim \mathcal{U}[0.785, 3.14] \times \mathcal{U}[0, 0.8]$ and generate trajectories by integrating for 200 steps ($\delta t = 0.05$) using a 4th order Runge-Kutta scheme, with an integration timestep $\delta t_{sim} = 0.001$. We split the dataset into train-validation-test subsets (50/25/25%). The train set is eventually obtained by extracting windows of 41 timesteps (with a stride of 2) from the original train split, yielding 8 100 smaller sub-trajectories.

For the data-driven, joint and alternate schemes, models are trained for 500 epochs using the Euler integrator with $\delta t = 0.05$. For the PD-based optimization scheme, each training consists of 50 epochs repeated 10 times, with the same integration settings. The final training of h_a consists of 150 epochs. All methods are optimized with Adam (Kingma et al., 2014).

4.B.3 REACTION-DIFFUSION SYSTEM

The reaction-diffusion system is a 2-dimensional system driven by the following PDEs (Klaasen et al., 1984):

$$\begin{cases} \frac{\delta u}{\delta t} &= \alpha \Delta u + R_u(u, v; \gamma) \\ \frac{\delta v}{\delta t} &= \beta \Delta v + R_v(u, v) \end{cases}$$

where $\mathbf{x} = (u, v)$ is the system state. $R_u(u, v; \gamma) = u - u^3 - \gamma - v$ and $R_v(u, v) = u - v$ are the local reaction terms. We choose a rectangular domain spanning $[-1, 1]^2$ with circular boundary conditions, that we discretize by a 32×32 uniform grid $G(t)$, where Δ is the 3×3 Laplace operator. For this system, (u, v) values are associated to each of the 32×32 grid cells G_{ij} that are governed by the same PDEs. Therefore, we decided to model $f_a = (R_u(u, v; \gamma), R_v(u, v))$ through a CNN with input $G(t)$.

For the generation of the dataset, we take inspiration from the work of Yin et al. (2021). We simulate a dataset of 1 920 trajectories spanning 246 timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 2.45]$ s. We first generate the initial states $\mathbf{x}_0 = (u_0, v_0) \sim \mathcal{U}[0, 1]^2$. We set $\alpha = 1e-3, \beta = 5e-3, \gamma = 5e-3$ and generate trajectories by integrating for 245 steps ($\delta t = 0.01$) using a 4th order Runge-Kutta scheme, with an integration timestep $\delta t_{sim} = 0.001$. We split the dataset into train-validation-test subsets (50/25/25%). The train set is eventually obtained by extracting windows of 51 timesteps (with a stride of 20) from the original train split, yielding 9 600 smaller sub-trajectories.

For the data-driven, joint and alternate schemes, models are trained for 1 000 epochs using the Euler integrator with $\delta t = 0.01$. For the PD-based optimization scheme, each training consists of 100 epochs repeated 10 times, with the same integration settings. The final training of h_a consists of 1 000 epochs. All methods are optimized with Adam (Kingma et al., 2014).

5 STEERING NEURAL NETWORK TRAINING THROUGH INTERPRETABLE CONSTRAINTS BASED ON PARTIAL DEPENDENCE

ADDENDUM

This chapter has previously been published as

Yann Claes, Pierre Geurts, and Vân Anh Huynh-Thu (2026). “Steering Neural Network Training through Interpretable Constraints Based on Partial Dependence”. In *arXiv preprint arXiv:2607.08641*. Submitted to TMLR.

Compared to the paper version, for the version presented in this chapter, we moved all the supplementary experiments into the main body. Furthermore, we aligned the notation with the other chapters.

In the previous chapters, we introduced a new training approach to learn hybrid additive models, based on partial dependence, for both static supervised regression problems and dynamical systems forecasting. On a range of examples, we have shown its positive impact in retrieving faithful first-principles models while yielding competitive hybrid models. In this chapter, we keep the focus on the concept of partial dependence but decide to tackle the problem of faithfulness from another angle. While hybrid additive methods incorporate domain knowledge into the inductive bias of models through the additive structure, the assumption of additivity can nevertheless be too restrictive and impair the performance of the final model as it is structurally encoded into the architecture. Instead, inspired by explanation-guided approaches, we propose a training method to guide model training by leveraging prior knowledge about model explanations, in the form of the partial dependence function on some inputs of interest.

After the introduction in Section 5.1, we quickly review related works in Section 5.2. Then, we present the scope of the problem as well as our network steering methodology in Section 5.3. Section 5.4 thoroughly studies our approach under different settings and we finally discuss our results and limitations in Section 5.5.

Contents

5.1	Introduction	59
5.2	Related work	59
5.3	Methodology	60
5.3.1	Partial dependence	60

5.3.2	Problem statement	61
5.3.3	Constraint configurations	63
5.4	Experiments	63
5.4.1	Friedman problem	64
5.4.2	Product problem	70
5.4.3	Concrete compressive strength prediction	73
5.4.4	PHALK dataset	74
5.4.5	Comparison to physics-informed models	76
5.4.6	Dynamical systems forecasting	82
5.4.7	Constraining with mis-specified knowledge	84
5.5	Conclusion	88
5.A	List of variables	93
5.A.1	Concrete compressive strength dataset	93
5.A.2	PHALK dataset	93

5.1 INTRODUCTION

Over the past decades, progress in machine learning (ML) methods has enabled practitioners in a variety of fields to tackle increasingly complex problems, in which such models serve as surrogates for first-principles models or costly simulators. While they come with great expressiveness, it is nevertheless not straightforward to interpret their reasoning process, which is why they are often referred to as "black box" models. There has been recently a vast development of the field of explainable artificial intelligence (XAI), in a desire to increase our understanding of decisions made by ML models (Adadi et al., 2018; Gilpin et al., 2018; Linardatos et al., 2020).

On the one hand, models can be made more interpretable, through specific design choices. For instance, Nelder et al. (1972) introduced generalized linear models (GLM) to model various response distributions, which are more flexible than linear regression, yet still very simple to explain. Hastie et al. (1986) later implemented generalized additive models (GAM) as an extension to GLMs with non-linear predictors, and Caruana et al. (2015) successfully applied GAMs to healthcare problems.

On the other hand, model predictions can be interpreted through post hoc (model-specific or model-agnostic) interpretation tools, with applications in different domains such as image processing (Q.-s. Zhang et al., 2018), healthcare (Vellido, 2020) and neural language processing (Madsen et al., 2022). For example, Lundberg et al. (2017) introduced SHAP values to compute feature importance in the prediction of the model. In a similar philosophy, Friedman (2001) designed partial dependence plots to visualize the marginal contribution of certain features to the model response, and complementary plotting functions were later implemented (Goldstein et al., 2015; Apley et al., 2020). For interpretations of visual predictions, most works rely on class activation mapping techniques to localize regions of importance (Zhou et al., 2016; Selvaraju et al., 2017; Chattopadhyay et al., 2018).

Although current interpretation tools can help domain experts to understand how decisions are made, they cannot take their feedback into account, which could be important if the model provides a correct decision for the wrong reasons. Gao, Gu, et al. (2024) notice that most works in XAI focus on the generation of model explanations, but leave aside the assessment of their quality, and judiciously state that adjusting models to produce more reasonable explanations is as important as producing them. For that purpose, several works have developed approaches such that, after training, model explanations align with specific domain knowledge (Ye Zhang et al., 2016; Yifei Zhang et al., 2024), a process known as explanation-guided learning (EGL). Furthermore, previous works in EGL have demonstrated that leveraging explanation knowledge could also help with limited training sizes (Gao, Sun, Bai, et al., 2022; Gao, Sun, Zhao, et al., 2022) and with out-of-distribution data (Taesiri et al., 2022).

In this work, we introduce a novel approach to control neural network training, based on partial dependence, such that its marginal response to certain inputs aligns with specific knowledge about the problem. We empirically validate the performance of our algorithm on several regression problems (including dynamical systems forecasting) and show that the obtained models perform better than their unconstrained counterparts. Moreover, they produce explanations faithful to prior knowledge, require less samples for training, and generalize better outside the originally considered input domain.

5.2 RELATED WORK

Explanation-guided learning is an emerging research field whose goal is to steer the training process of ML models to improve their explainability, according to specific

domain knowledge. Several works implement these principles, mostly for knowledge in the form of importance of input features and input regions (Rieger et al., 2020; Yifei Zhang et al., 2024), or in the form of global properties that the model should satisfy, such as sparsity (Wu et al., 2020), or even to mitigate bias (Liu et al., 2019). In comparison with previous works, our approach rather assumes a fundamentally different type of knowledge, in the form of a specific function of a subset of features, i.e. the partial dependence. Furthermore, while previous works focus on classification tasks based on images or textual data, our approach rather focuses on regression tasks applied with tabular data or dynamical systems.

ML models can be trained specifically to satisfy known constraints about the problem. For example, convexity (Pya et al., 2015; Amos et al., 2017) and monotonicity (Pya et al., 2015; Wehenkel et al., 2019) constraints can be enforced on neural networks or other types of models. Structural constraints can also be required directly on the output of the models (Hendriks et al., 2020; A. Chen et al., 2023). Nevertheless, for all these approaches, constraints are imposed on the function itself, not on the explanations.

Another related line of work is that of physics-informed machine learning. In this research area, physics-based models are combined with ML models to enhance physical consistency and improve generalization. In practice, most hybrid models sum the contributions of the first-principles model and of the ML model (Qian et al., 2021; Takeishi et al., 2021). This decomposition implies finding the right balance between the physics-based and ML models, which has been investigated in several works (Yin et al., 2021; Donà et al., 2022; Claes, Huynh-Thu, et al., 2025). In opposition to such approaches, our method is less restrictive as it does not assume additivity or any other form of model hybridization, but simply relies on a single ML model. Furthermore, these training algorithms do not take into account the explanations of the resulting models, hence frameworks are radically different.

5.3 METHODOLOGY

In this section, we first provide some mathematical background and intuition about the partial dependence as a model explanation. We then formally define the class of problems that is studied, along with our proposed training algorithm and the different configurations that we consider. We focus here on standard regression problems. Section 5.4.6 explains how the approach can be extended to dynamical systems forecasting.

5.3.1 PARTIAL DEPENDENCE

Let us define the regression problem $y = f(\mathbf{x}) + \varepsilon$, where $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^d$, with $d \in \mathbb{N}_+$, drawn from a distribution $p(\mathbf{x}, y)$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ some Gaussian noise.

Definition 1. Let $\mathbf{x}_k$ be some subset of features of interest and $\mathbf{x}_{-k}$ its complement, with $\mathbf{x}_k \cup \mathbf{x}_{-k} = \mathbf{x}$, then the partial dependence (PD) of a function $f(\mathbf{x})$ on $\mathbf{x}_k$ is defined as (Friedman, 2001):

$$\begin{aligned} PD_\infty(f, \mathbf{x}_k) &= \mathbb{E}_{\mathbf{x}_{-k}} [f(\mathbf{x}_k, \mathbf{x}_{-k})], \\ &= \int f(\mathbf{x}_k, \mathbf{x}_{-k}) p(\mathbf{x}_{-k}) d\mathbf{x}_{-k}, \end{aligned} \quad (5.1)$$

with $p(\mathbf{x}_{-k})$ the marginal distribution of $\mathbf{x}_{-k}$.

In practice, with a finite learning set $LS = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$ of N samples, the PD is estimated by

$$PD_N(f, \mathbf{x}_k) = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_k, \mathbf{x}_{-k}^{(i)}). \quad (5.2)$$

In other words, the PD measures how a certain subset of features $\mathbf{x}_k$ impacts the output of a function f , on average. Now, let us assume that the entire effect of $\mathbf{x}_k$ is known algebraically as $f_k(\mathbf{x}_k)$, some function of $\mathbf{x}_k$. If such effect is purely additive, i.e. $f(\mathbf{x}) = f_k(\mathbf{x}_k) + f_{-k}(\mathbf{x}_{-k})$, where $f_{-k}(\mathbf{x}_{-k})$ is an unknown function of $\mathbf{x}_{-k}$, then

$$PD_\infty(f, \mathbf{x}_k) = f_k(\mathbf{x}_k) + \mathbb{E}_{\mathbf{x}_{-k}} [f_{-k}(\mathbf{x}_{-k})].$$

If the effect of $\mathbf{x}_k$ is instead multiplicative, i.e. $f(\mathbf{x}) = f_k(\mathbf{x}_k) \cdot f_{-k}(\mathbf{x}_{-k})$, then it follows

$$PD_\infty(f, \mathbf{x}_k) = f_k(\mathbf{x}_k) \cdot \mathbb{E}_{\mathbf{x}_{-k}} [f_{-k}(\mathbf{x}_{-k})].$$

The PD is thus straightforward to compute in both cases, and we can write

$$PD_\infty(f, \mathbf{x}_k) = \phi_0^* \cdot f_k(\mathbf{x}_k) + \phi_1^*, \quad (5.3)$$

where $\phi_0^* \in \mathbb{R}$ and $\phi_1^* \in \mathbb{R}$ are scaling and bias constants. To illustrate, let us consider the *Friedman problem* (Friedman et al., 1983), which is used in our experiments below:

$$f(\mathbf{x}) = \psi_0 \sin(\psi_1 x_0 x_1) + \psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4.$$

Taking $\mathbf{x}_k = \{x_0, x_1\}$, $\mathbf{x}_{-k} = \{x_2, x_3, x_4\}$, it follows

$$PD_\infty(f, \mathbf{x}_k) = \phi_0^* \cdot \psi_0 \sin(\psi_1 x_0 x_1) + \phi_1^*,$$

where $\phi_0^* = 1$, $\phi_1^* = \mathbb{E}_{\mathbf{x}_{-k}} [\psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4]$.

5.3.2 PROBLEM STATEMENT

We assume knowledge about model explanations in the form of a PD function, and are given a fixed learning set $LS = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$ of N samples. Formally, let us assume some differentiable ML model $h^\theta(\mathbf{x})$ where $\theta \in \mathbb{R}^{d_\theta}$ is a trainable parameter vector. We want to learn θ such that

$$\theta^* = \arg \min_{\theta} \mathcal{L}_{mse}(h^\theta, f; LS) \quad (5.4)$$

$$\text{s.t. } PD_N(h^\theta, \mathbf{x}_k) = PD_N(f, \mathbf{x}_k), \quad \forall \mathbf{x}_k \in \mathbb{R}^{d_{\mathbf{x}_k}} \quad (5.5)$$

where

$$\mathcal{L}_{mse}(h^\theta, f; LS) = \frac{1}{N} \sum_{i=1}^N (h^\theta(\mathbf{x}^{(i)}) - y^{(i)})^2. \quad (5.6)$$

Equation (5.5) assumes perfect knowledge of the PD up through the values of its parameters, which could be unrealistic in some cases. We hence assume that we do not have a perfect knowledge of the PD, but rather of the algebraic shape that the PD should have. Therefore, we introduce a function $h_k^\phi(\mathbf{x}_k)$, where $\phi \in \mathbb{R}^{d_\phi}$ is a learnable parameter vector including the previously defined scaling and bias parameters ϕ_0 and ϕ_1 , such that for the optimal vector ϕ^* , we have $h_k^{\phi^*}(\mathbf{x}_k) = PD_\infty(f, \mathbf{x}_k)$. Taking back the previous example yields $h_k^\phi(\mathbf{x}_k) = \phi_0 \cdot [\phi_2 \sin(\phi_3 x_0 x_1)] + \phi_1$, and $\phi^* = [1, C, \psi_0, \psi_1]$ would be a potential optimal vector, with $C = \mathbb{E}_{\mathbf{x}_{-k}} [\psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4]$.

As Equation (5.5) is hard to satisfy tightly, we rely on the following relaxation, introducing a coefficient $\lambda \in [0, 1]$

$$\theta^*, \phi^* = \arg \min_{\theta, \phi} \mathcal{L}_\lambda(h^\theta, h_k^\phi, f; LS), \quad (5.7)$$

where

$$\mathcal{L}_\lambda(h^\theta, h_k^\phi, f; LS) = (1 - \lambda) \mathcal{L}_{mse}(h^\theta, f; LS) + \lambda \mathcal{L}_{PD}(h^\theta, h_k^\phi; LS).$$

The second term is defined as

$$\mathcal{L}_{PD}(h^\theta, h_k^\phi; LS) = \frac{1}{N} \sum_{i=1}^N (PD_N(h^\theta, \mathbf{x}_k^{(i)}) - h_k^\phi(\mathbf{x}_k^{(i)}))^2. \quad (5.8)$$

In practice, we apply steps of gradient descent to fit both θ and ϕ in Equation (5.7), for a certain number of epochs E . Although fitting both θ and ϕ could be performed jointly, we decide to alternate both steps. To initialize the process, we arbitrarily fit h_k^ϕ for E epochs on the raw outputs, yielding

$$\phi^{(0)} = \arg \min_{\phi} \mathcal{L}_{prior}(h_k^\phi, f; LS), \quad (5.9)$$

where $\mathcal{L}_{prior}(h_k^\phi, f; LS) = \frac{1}{N} \sum_{i=1}^N (h_k^\phi(\mathbf{x}_k^{(i)}) - y^{(i)})^2$. Then, we fit the ML model under the constraint that its PD should be close to the current model $h_k^{\phi^{(0)}}$. Every $S \in \mathbb{N}^+$ epochs, we update ϕ to minimize Equation (5.8), for the current fit of θ . We further repeat the previous two steps.

It is important to note that, given the definition of Equation (5.8), we can compute $\mathcal{L}_{PD}$ using a dataset $LS_{PD} = \{\mathbf{x}_k^{(j)}\}_{j=1}^{N_{PD}}$ different from LS :

$$\mathcal{L}_{PD}(h^\theta, h_k^\phi; LS, LS_{PD}) = \frac{1}{N_{PD}} \sum_{j=1}^{N_{PD}} (PD_N(h^\theta, \mathbf{x}_k^{(j)}) - h_k^\phi(\mathbf{x}_k^{(j)}))^2.$$

While LS is used to compute the value of PD_N at a given point (i.e. to obtain samples from $p(\mathbf{x}_{-k})$), LS_{PD} is a dataset of said given points $\mathbf{x}_k$. The utility of constructing separate datasets for these two steps is further explained in Section 5.3.3. Algorithm 7 summarizes the training procedure, where steps of gradient descent ∇ are performed on mini-batches.

Algorithm 7 Partial Dependence Steering

Input: $LS = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N, LS_{PD} = \{\mathbf{x}_k^{(j)}\}_{j=1}^{N_{PD}}$
Parameters: $E > S \geq 0, \tau_1 > 0, \tau_2 > 0, \lambda > 0$

for $e = 1$ **to** E **do**
 $\phi^{(e)} \leftarrow \phi^{(e-1)} - \tau_1 \nabla_{\phi} \mathcal{L}_{prior}(h_k^{\phi^{(e-1)}}, f; LS)$
 $\phi^{(0)} \leftarrow \arg \min_e \mathcal{L}_{prior}(h_k^{\phi^{(e)}}, f; LS)$

$s = 0, r = 1$
for $e = 1$ **to** E **do**
 if $s < S$ **then**
 $\theta^{(e)} \leftarrow \theta^{(e-1)} - \tau_2 \nabla_{\theta} \mathcal{L}_{\lambda}(h^{\theta^{(e-1)}}, h_k^{\phi^{(r-1)}}, f; LS, LS_{PD})$
 $s = s + 1$
 else
 $\phi^{(r)} \leftarrow \phi^{(r-1)} - \tau_2 \nabla_{\phi} \mathcal{L}_{PD}(h^{\theta^{(e-1)}}, h_k^{\phi^{(r-1)}}; LS, LS_{PD})$
 $\theta^{(e)} \leftarrow \theta^{(e-1)}$
 $s = 0, r = r + 1$
 $\theta \leftarrow \arg \min_e \mathcal{L}_{mse}(h^{\theta^{(e)}}, f; LS)$
return θ

5.3.3 CONSTRAINT CONFIGURATIONS

We consider different scenarios, where we vary both the knowledge level and LS_{PD} . In all cases, the shape of $PD_\infty(f, \mathbf{x}_k)$ is assumed to be known through $h_k^\phi(\mathbf{x}_k)$. However, the values of the parameters ϕ can either be known exactly (i.e., $\phi = \phi^*$, denoted *ground truth prior*), or have to be learned (i.e., $\phi = \hat{\phi}$, denoted *approximated prior*), as in Algorithm 7. Furthermore, we can define LS_{PD} in two ways: $LS_{PD} = \{\mathbf{x}_k^{(i)} \mid \mathbf{x}^{(i)} \in LS\}_{i=1}^N$ (*in-distribution constraint*), where the PD of the model will be constrained at positions defined by the learning sample, i.e. $\mathbf{x}_k \sim p(\mathbf{x}_k)$, or as a set of evenly spaced values between lower and upper bounds $l, u \in \mathbb{R}^{d_{\mathbf{x}_k}}$, i.e. $LS_{PD} = \{\mathbf{x}_k^{(j)} \in \mathbb{R}^{d_{\mathbf{x}_k}} \mid \mathbf{x}_k^{(j)} = l + (j-1) \frac{u-l}{N_{PD}-1}\}_{j=1}^{N_{PD}}$ (*linspace constraint*). Such bounds can be the minimum and maximum observed values for $\mathbf{x}_k$ in the dataset, or derived from prior knowledge. Constructing a separate dataset LS_{PD} allows to constrain on regions that may be poorly sampled, thereby helping the model to generalize more effectively. In particular, this could help in a scenario where the model is later on fed with out-of-distribution data.

5.4 EXPERIMENTS

In this section, we study the performance of our PD-based steering method, in terms of both predictive performance and quality of the resulting PD, through test set estimates of Equation (5.6) and Equation (5.8), with $\phi = \phi^*$, which we refer to as $\mathcal{L}_{mse}$ and $\mathcal{L}_{PD}$. We compare our method to unconstrained training on synthetic and real regression problems, as well as on a problem of dynamical systems forecasting. We study the effect of training sample size on both measures, and analyze the impact of shifting the domain of $\mathbf{x}_k$ at test time. We also compare the performance of our method against that of physics-informed ML models, and eventually study the behavior of models constrained with mis-specified prior knowledge.

Dataset and model hyperparameters For synthetic regression problems, we generate training and validation sets of successively 300, 150 and 50 samples each. All methods are evaluated on a fixed test set of 10 000 samples. For the lower and upper bounds of linspace constraints, we use the bounds of the input domain, and we arbitrarily fix the resolution to 25 for PD functions of two variables (producing a 25×25 grid) and to 256 for single-variable PD functions. For both constraining types, we draw random mini-batches from LS_{PD} at each epoch. We consider the following settings (see Section 5.3.3):

- Linspace constraint on the ground truth prior, denoted as $\phi = \phi^* (L)$,
- In-distribution constraint on the ground truth prior, denoted as $\phi = \phi^* (ID)$,
- Linspace constraint on the approximated prior, denoted as $\phi = \hat{\phi} (L)$,
- In-distribution constraint on the approximated prior, denoted as $\phi = \hat{\phi} (ID)$.

We fix the architecture of the neural network to three layers of increasing hidden size (32, 64 and 128), with a fixed training batch size of 16 samples. We optimize the learning rate τ_2 in $\{5 \times 10^{-2}, 5 \times 10^{-3}, 5 \times 10^{-4}\}$ and the coefficient λ in $\{0.25, 0.5, 0.75\}$, selecting the values that minimize $\mathcal{L}_{mse}$ on the validation set. We set the number of epochs $E = 1000$, $\tau_1 = 1$, $S = 50$.

Test-time domain shift To implement this behavior, in synthetic problems, we first define an interpolation range on a particular variable in $\mathbf{x}_k$. We then generate training and validation sets by sampling from the original distribution $p(\mathbf{x})$, rejecting any sample whose value for the selected variable lies outside the aforementioned range. The test set

is identical for interpolation and extrapolation, such that it contains samples with values both inside and outside the interpolation range.

5.4.1 FRIEDMAN PROBLEM

We consider the following problem (Friedman et al., 1983):

$$y = \psi_0 \sin(\psi_1 x_0 x_1) + \psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4 + \sum_{j=5}^9 0x_j + \varepsilon,$$

where $x_j \sim \mathcal{U}(0, 1)$, $j = 0, \dots, 9$, $\psi = [10, \pi, 20, 0.5, 10, 5]$ and $\varepsilon \sim \mathcal{N}(0, 1)$. In what follows, we consider three experiments constraining the PD of h^θ w.r.t. the following functions, respectively:

$$\begin{aligned} h_k^\phi(x_0, x_1) &= \phi_0 [\phi_2 \sin(\phi_3 x_0 x_1)] + \phi_1, \\ h_k^\phi(x_2) &= \phi_0 [\phi_2 (x_2 - \phi_3)^2] + \phi_1, \\ h_k^\phi(x_3, x_4) &= \phi_0 [\phi_2 x_3 + \phi_3 x_4] + \phi_1. \end{aligned}$$

Potential ground truth parameter vectors can be respectively:

$$\begin{aligned} \phi^* &= [1, C_1, \psi_0, \psi_1], & C_1 &= \mathbb{E}_{\mathbf{x}_{-k}} [\psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4], \\ \phi^* &= [1, C_2, \psi_2, \psi_3], & \text{where } C_2 &= \mathbb{E}_{\mathbf{x}_{-k}} [\psi_0 \sin(\psi_1 x_0 x_1) + \psi_4 x_3 + \psi_5 x_4], \\ \phi^* &= [1, C_3, \psi_4, \psi_5], & C_3 &= \mathbb{E}_{\mathbf{x}_{-k}} [\psi_0 \sin(\psi_1 x_0 x_1) + \psi_2 (x_2 - \psi_3)^2]. \end{aligned}$$

CONSTRAINING $PD_N(h^\theta, x_0, x_1)$

Sample size effect As a general trend, we observe in Figure 5.1 that decreasing the learning sample size degrades both measures of performance, for all methods. Furthermore, constraining the PD of the model improves both measures compared to unconstrained training, for all training sizes.

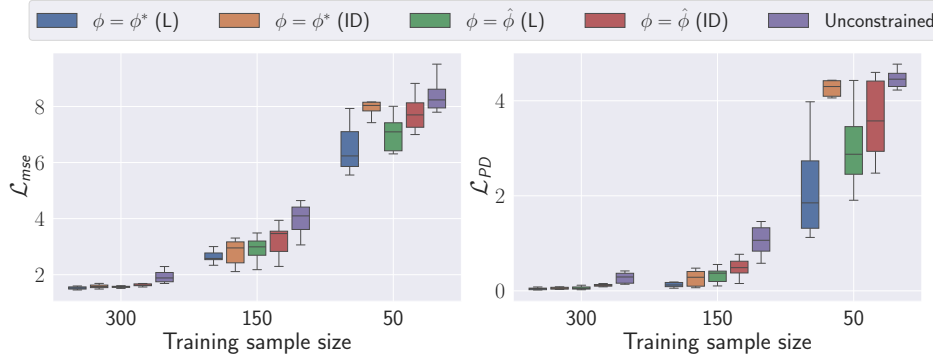


Figure 5.1. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_0, x_1)$. Each boxplot summarizes the results on the test set, over 10 different training initializations.

As could be expected, linspace constraining on the true PD function (blue boxes) achieves the best results for both metrics, whatever the sample size. Nevertheless, it is interesting to note that linspace constraining with an approximated prior $\phi = \hat{\phi}$ (green boxes) performs on par with in-distribution constraining on true PD samples (orange boxes), and even outperforms the latter when data becomes very scarce (50 samples), which highlights the benefit of constraining the PD in poorly-sampled regions of the input space. This can be verified by looking at Figure 5.2, which highlights the difference in the PD estimates.

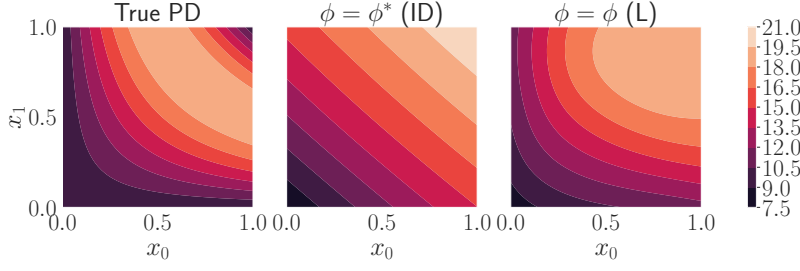


Figure 5.2. Examples of PD estimation for $PD_N(h^\theta, x_0, x_1)$. The figure displays the true PD (left), the PD after in-distribution constraining on the ground truth PD (middle), and the PD after linspace constraining on the approximated prior (right). Both have been trained with 50 samples, in the interpolation setting.

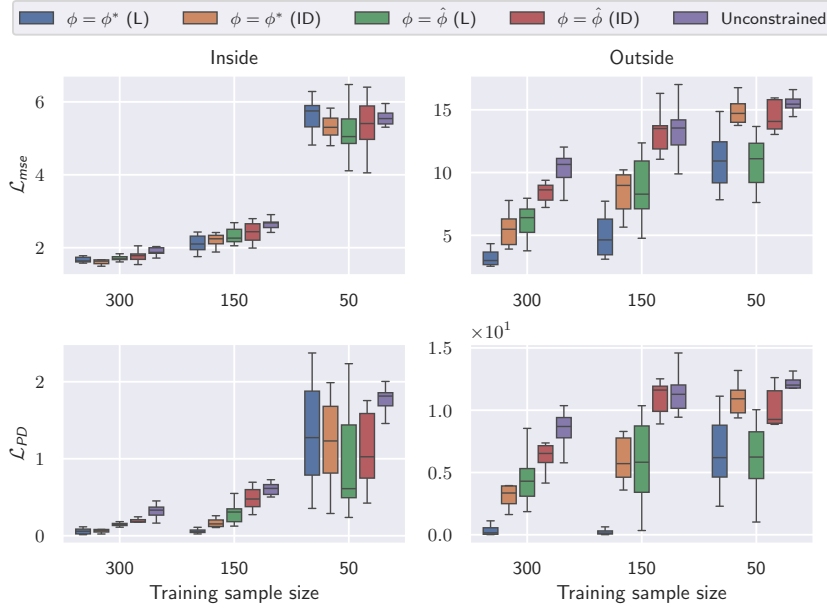


Figure 5.3. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_0, x_1)$. *Inside* metrics are computed on samples for which $x_0 \leq 0.75$ (left plots), while *Outside* metrics are computed on samples where $x_0 > 0.75$ (right plots). Each boxplot summarizes the results on the test set, over 10 different training initializations.

Test-time domain shift We further test the robustness of constrained models when observing test-time domain shift in $\mathbf{x}_0$: samples for which $\mathbf{x}_0 > 0.75$ are rejected from the training and validation sets. We can observe in Figure 5.3 that the relative ranking of training methods inside the interpolation range is similar to that observed in Figure 5.1. Note that this similarity is not necessarily expected as training settings are quite different from the previous experiment. Indeed, for in-distribution constraints, given that the PD of the model is now constrained only up to $x_0 = 0.75$, the model can produce different results from the one constrained on the entire domain, even inside the interpolation range. Looking at $\mathcal{L}_{mse}$ inside the interpolation range, we notice that we now obtain overall lower errors compared to the previous setting, whatever the training scheme. This is expected since training methods now have access to more data samples in a narrower space, but it could also mean that the output signal for $x_0 > 0.75$ is more complex to

capture. For both metrics, PD steering helps to dampen the impact of moving outside the interpolation range, once again much more when constraining on evenly-spaced points.

CONSTRAINING $PD_N(h^\theta, x_2)$

Sample size effect We can observe in Figure 5.4 that constraining $PD_N(h^\theta, x_2)$ helps to improve both measures of performance, for each training size, and it now seems that linspace constraining and in-distribution constraining are mostly on par, which was not the case when constraining on $PD_N(h^\theta, x_0, x_1)$. This may be explained by the fact that the PD function is now only one-dimensional and data points are uniformly sampled in $[0, 1]$, meaning that the input domain on x_2 is more densely covered than that of two-dimensional PD functions. Furthermore, we also hypothesize that both constraining methods (i.e., linspace and in-distribution) perform similarly because the PD shape is symmetrical and less intricate to approximate.

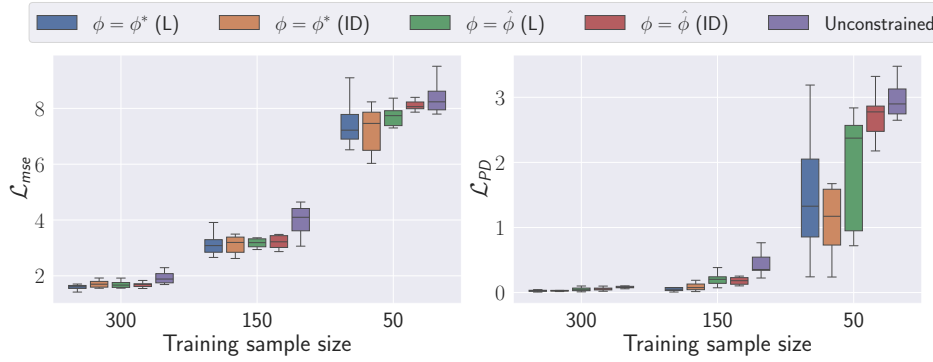


Figure 5.4. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_2)$. Each boxplot summarizes the results over 10 different training initializations.

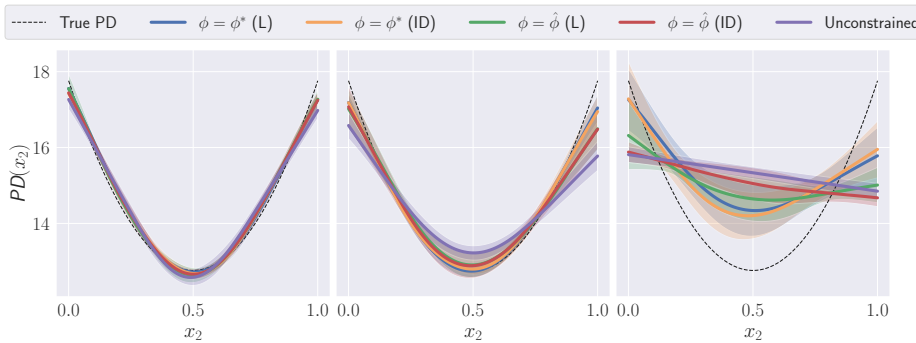


Figure 5.5. Mean PD estimations for $PD_N(h^\theta, x_2)$ obtained with each method, for the Friedman problem, in the interpolation setting. All methods have been trained respectively on 300 samples (left), 150 samples (middle), 50 samples (right). Mean and standard deviations are computed over 10 different training initializations.

Figure 5.5 displays the average PD estimations obtained when constraining $PD_N(h^\theta, x_2)$ for each training size, in interpolation. We can observe that with enough data points (300), all methods (including unconstrained training) are able to provide reliable estimates. With half as many training samples, unconstrained PD estimates are now less precise, whereas with 50 training samples, it produces a straight curve. We also note that for

the latter size, all training methods seem to shift the PD estimates upwards, which we attribute to the initial objective of minimizing Equation (5.6) overtaking the minimization of Equation (5.8), as only the former objective is used as model selection criterion.

Test-time domain shift We now enforce test-time domain shift in x_2 . As was the case for the previous Friedman experiment, samples for which $x_2 > 0.75$ are rejected from the training and validation sets.

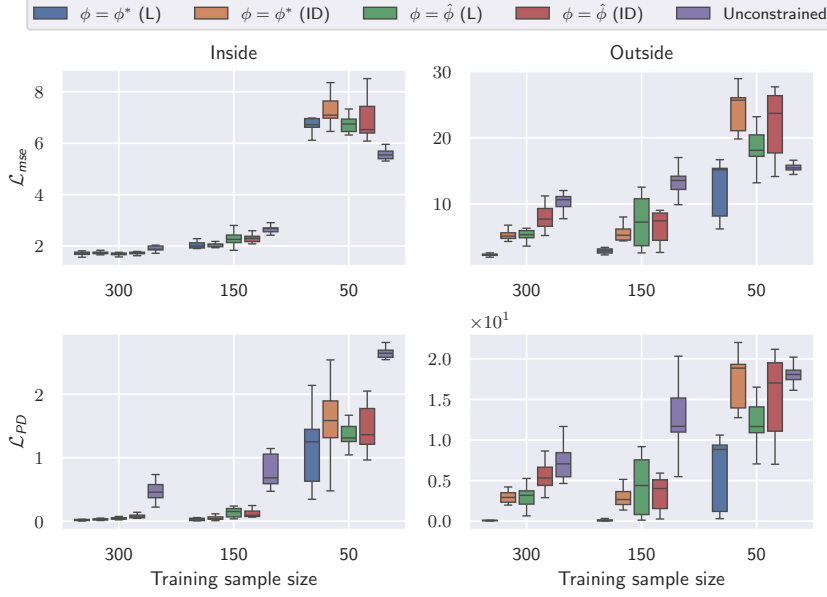


Figure 5.6. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_2)$. *Inside* metrics are computed on samples for which $x_2 \leq 0.75$ (left plots), while *Outside* metrics are computed on samples where $x_2 > 0.75$ (right plots). Each boxplot summarizes the results on a test set, over 10 different training initializations.

In Figure 5.6, we observe very similar results to those obtained in Figure 5.4, where constrained training improves both measures of performance. Nevertheless, we notice that after training on 50 samples, estimations of the PD function for both unconstrained training (purple boxes) and constrained training with an approximated prior (green and red boxes) seem better than previously, by simply looking at Figure 5.6 (left). This can however be explained by the fact that the target domain is now cut at $x_2 = 0.75$ for *inside* estimates, therefore omitting an important part where larger error amplitudes are concentrated. This can be further verified by analyzing Figure 5.7 (right). Indeed, we can see that linspace constraining on the ground truth PD function yields the least worse estimations, as it is the only method that has access to high-quality information about the PD outside the interpolation domain. Even though the optimal in-distribution constraining (orange curve) also has access to the same level of quality, it is nonetheless limited up to $x_2 = 0.75$. As far as constraining with approximated priors is concerned (green and red curves), neither method produces good PD estimates as they are limited by the quality of the prior, which is only fitted inside the interpolation range. Furthermore, due to the limited amount of training points, it seems that, even in the interpolation range, models put more weight to the minimization of the total mean-squared error rather than the minimization of the distance to the (true) target PD function.

For comparison, we display in Figure 5.7 (left) the PD estimations obtained in

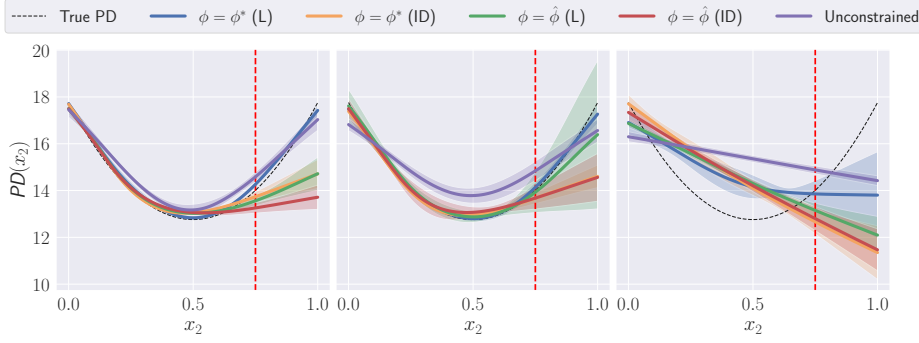


Figure 5.7. Mean PD estimations for $PD_N(h^\theta, x_2)$ obtained with each method, for the Friedman problem, in the extrapolation setting. All methods have been trained respectively on 300 samples (left), 150 samples (middle), 50 samples (right). Mean and standard deviations are computed over 10 different training initializations. The dotted red line represents the boundary of the interpolation range.

extrapolation, after training with 300 samples. We can observe that now all methods provide quite reliable PD estimates in the interpolation domain, which was not the case with only 50 samples. Furthermore, linspace constraining on the true PD function (blue curve) vastly outperforms other baselines, as expected. It is interesting to notice that, in this scenario, linspace constraining (green curve) produces PD estimations similar to in-distribution constraining on the true PD function (orange curve). This means that, even with imperfect knowledge about the latter, it still seems better to constrain on a wider region. Even more interestingly, while estimations after training on 150 samples show more variance, it seems that mean PD estimations of such linspace fitting are closer to the true PD function than mean estimations obtained on 300 samples. PD estimates obtained from unconstrained training clearly degrade as training size decreases.

CONSTRAINING $PD_N(h^\theta, x_3, x_4)$

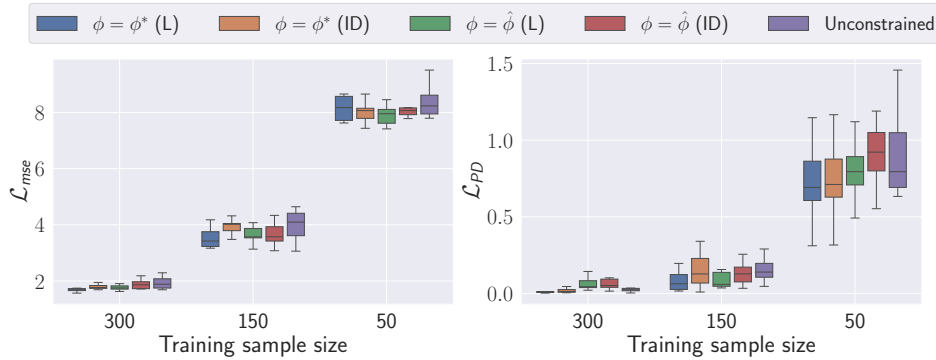


Figure 5.8. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_3, x_4)$. Each boxplot summarizes the results over 10 different training initializations.

Sample size effect While the impact of PD constraining was quite marked for $PD_N(h^\theta, x_0, x_1)$ and $PD_N(h^\theta, x_2)$, constraining on $PD_N(h^\theta, x_3, x_4)$ seems to have very little impact on both the total mean-squared error and on the estimated PD, as can be seen in Figure 5.8. In fact, we notice that the scale of the PD error is much smaller than that of the two previous experiments, floating around 0.8 for 50 samples, whereas it is twice to five times as large for $PD_N(h^\theta, x_0, x_1)$ and $PD_N(h^\theta, x_2)$. This null impact of PD

constraining can be explained by the fact that $h_k^\phi(x_3, x_4) = \phi_0[\phi_2 x_3 + \phi_3 x_4] + \phi_1$ is actually very easy to capture from the output signal. To illustrate this claim, we provide in Figure 5.9 samples of PD estimations for $PD_N(h^\theta, x_0, x_1)$ (top) and $PD_N(h^\theta, x_3, x_4)$ (bottom), after training with 50 samples. It is easy to notice, for the latter, that even unconstrained training yields a PD estimation that is close to the real one, which is not true for the former.

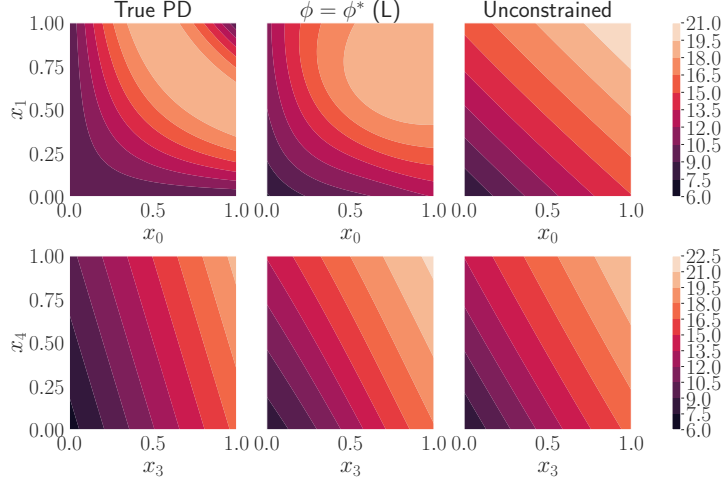


Figure 5.9. Example of PD estimation for $PD_N(h^\theta, x_0, x_1)$ (top plots) and $PD_N(h^\theta, x_3, x_4)$ (bottom plots). The figure displays the true PD (left), the PD after linspace constraining on the ground truth PD (middle), and the PD after unconstrained training (right). Both have been trained with 50 samples, in the interpolation setting.

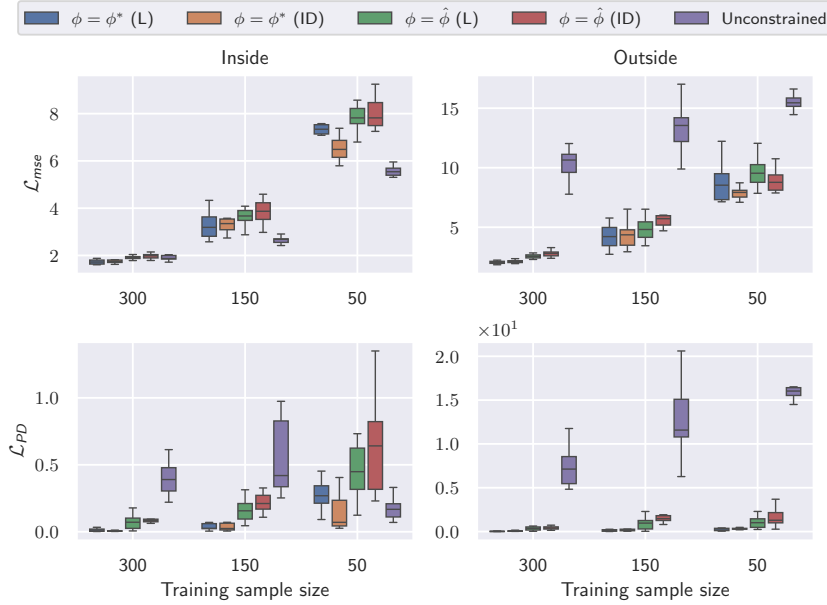


Figure 5.10. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_3, x_4)$. *Inside* metrics are computed on samples for which $x_3 \leq 0.75$ (left plots), while *Outside* metrics are computed on samples where $x_3 > 0.75$ (right plots). Each boxplot summarizes the results on a test set, over 10 different training initializations.

Test-time domain shift As for $PD_N(h^\theta, x_0, x_1)$, samples for which $x_3 > 0.75$ are rejected from the training and validation sets.

Similarly to the interpolation setting, we can observe in Figure 5.10 that constraining on $PD_N(h^\theta, x_3, x_4)$ slightly helps in extrapolation, in terms of total mean-squared error. Nevertheless, the resulting PD estimations after constraining with the ground truth prior now seem to be consistently better than unconstrained training, which can be verified in Figure 5.11. Even though both training methods captured the shape of the underlying PD function, unconstrained training fails to approximate accurately the output scale, especially outside the interpolation range, i.e. where $x_3 > 0.75$.

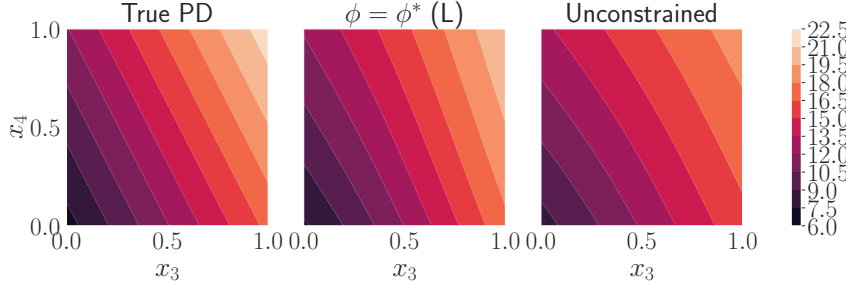


Figure 5.11. Example of PD estimation for $PD_N(h^\theta, x_3, x_4)$. The figure displays the true PD (left), the PD after linspace constraining on the ground truth PD (middle), and the PD after unconstrained training (right). Both have been trained with 50 samples, in the extrapolation setting.

5.4.2 PRODUCT PROBLEM

We consider the following problem:

$$y = \psi_0 x_0^3 \cos(\psi_1 x_0) \frac{(x_1 - 1)^2}{2} + \sum_{j=2}^9 0x_j + \varepsilon,$$

where features are correlated with zero mean, variances and covariances equal to 0.5 and 0.25, respectively, $\psi = [15, \pi]$, and $\varepsilon \sim \mathcal{N}(0, 1)$, with the following PD function:

$$h_k^\phi(x_0) = \phi_0 [\phi_2 x_0^3 \cos(\phi_3 x_0)] + \phi_1.$$

A potential optimal parameter vector is $\phi^* = [C, 0, \psi_0, \psi_1]$, with $C = \mathbb{E}_{x_1} \left[\frac{(x_1 - 1)^2}{2} \right]$.

Sample size effect As was the case for the Friedman problem, we can observe in Figure 5.12 that model performance degrades as the training size decreases, although more slightly. We notice that our regularization methods vastly help to learn more robust models with limited amounts of data compared to unconstrained training. Indeed, the test mean-squared errors of in-distribution constrained models (orange and red boxes) on 50 samples are roughly at the same level as those of the unconstrained model trained on 300 samples, and linspace constrained models (blue and green boxes) vastly outperform the latter. We also notice that the PD obtained for most constrained models on 50 samples is even better than that of the unconstrained neural network on 300 samples. Figure 5.12 also highlights the benefits of constraining on a set of evenly-spaced points, as a wider region of the input domain is covered, even against in-distribution constraining on the ground truth PD function (orange boxes). Interestingly, linspace constraining on an approximated prior (green boxes) can produce more accurate predictions compared to linspace constraining on the true function (blue boxes). This can be explained by the fact

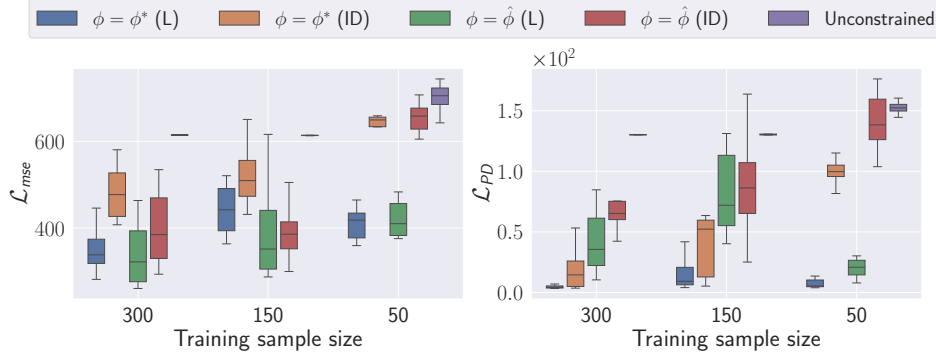


Figure 5.12. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the product problem, constraining $PD_N(h^\theta, x_0)$. Each boxplot summarizes the results on the test set, over 10 different training initializations.

that, by approximating the PD function, the former allows for more flexibility to reduce the mean-squared error on the outputs, at the cost of a worse PD estimation.

We represent in Figure 5.13 (left) the different PD estimations obtained on average with each training method, with a training set of 150 samples. We notice that each constrained model approximates the true PD function quite well, while the unconstrained neural network has more difficulties in approximating the boundary regions, where the PD function varies more sharply.

Even though PD plots (PDP) as interpretation tools are expected to deteriorate and be less faithful in the presence of (strongly) correlated features, due to the averaging of predictions over possibly unrealistic input samples (Molnar, 2020), we observe that this does not impact our method as we constrain models with the true algebraic form of the PD.

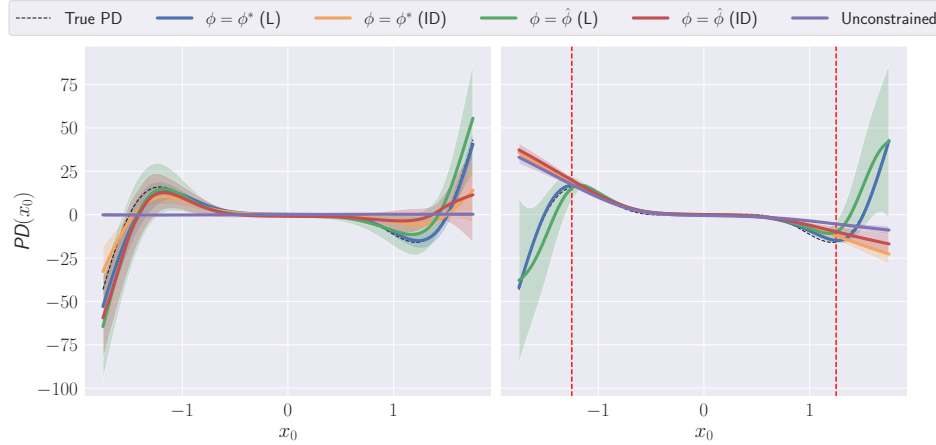


Figure 5.13. Mean PD estimations for $PD_N(h^\theta, x_0)$ obtained with each method, for the product problem, in the interpolation (left) and extrapolation settings (right), after training on 150 samples. Mean and standard deviations are computed over 10 different training initializations. The dotted red lines represent the boundaries of the interpolation range.

In Figure 5.14, we show that in-distribution constraining fails to provide an accurate PD estimate, compared to linspace constraining, when the number of training points is scarcer ($N = 50$).

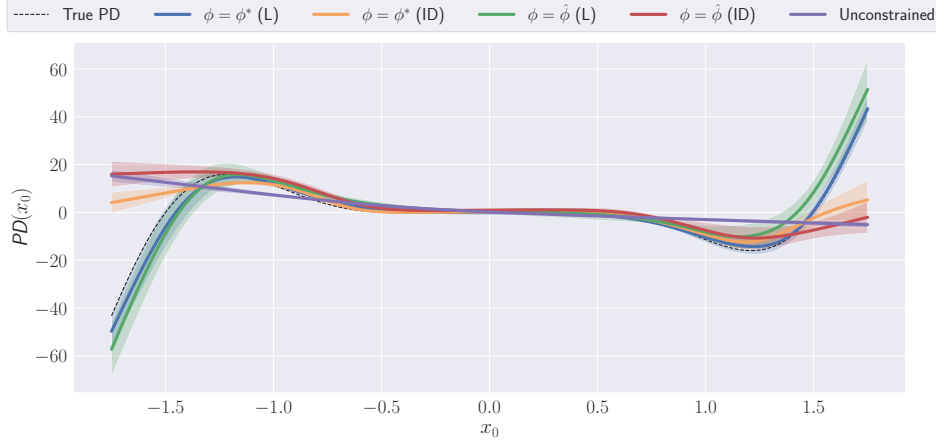


Figure 5.14. Mean PD estimations for $PD_N(h^\theta, x_0)$ obtained with each method, for the product problem, in the interpolation setting. All methods have been trained on 50 samples. Mean and standard deviations are computed over 10 different training initializations.

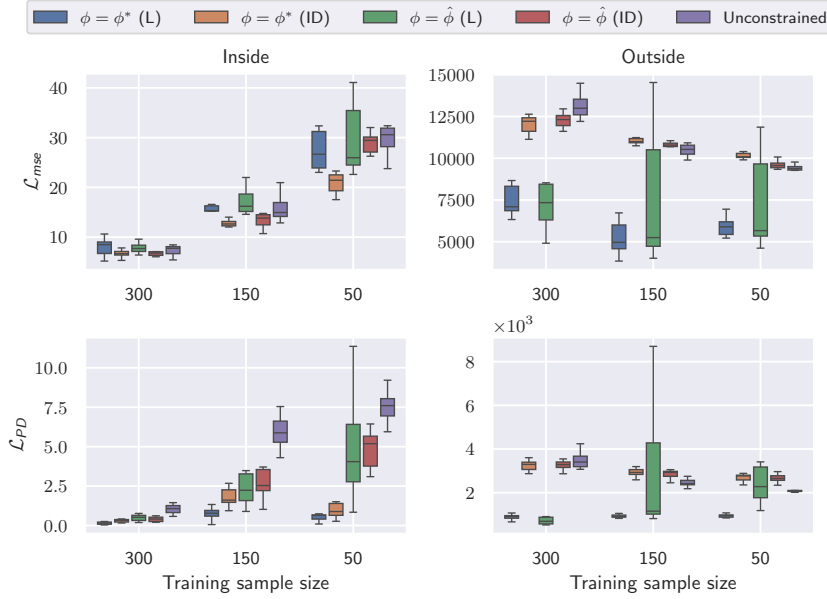


Figure 5.15. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the product problem, constraining $PD_N(h^\theta, x_0)$. *Inside* metrics are computed on samples for which $x_0 \in [-1.25, 1.25]$ (left plots), while *Outside* metrics are computed on samples outside that range (right plots). Each boxplot summarizes the results on the test set, over 10 different training initializations.

Test-time domain shift We study the impact of shifting the domain of x_0 at test time: samples for which $x_0 \notin [-1.25, 1.25]$ are rejected from the training and validation sets, which roughly corresponds to the switch of concavity of the PD function, on both sides. In Figure 5.15, we can observe that for training sets of 300 and 150 samples, unconstrained training performs on par (if not sometimes better) with constrained methods, when samples lie inside the interpolation range. By comparing with the mean-squared errors reported in Figure 5.12, we hypothesize that the output region in the center of the domain

is easier to approximate than the borders, which is also suggested by the true PD function shape in Figure 5.14. This claim can be further verified by looking at both metrics outside the interpolation range. Indeed, linspace-constrained models (blue and green boxes) vastly outperform unconstrained ones in both measures, producing similar results whatever the training set size, although with some variance when the PD parameters need to be estimated. On the other hand, it is interesting to note that in-distribution constraining (orange and red boxes) only slightly improves performance (or even worsens) against unconstrained training. This can be again explained by the fact that the PD of the model is constrained only inside the interpolation domain, while most modeling complexity lies at the boundaries of the domain, which is what we verify in Figure 5.13 (right). Indeed, the PD estimations of in-distribution training are actually very similar to those obtained after unconstrained training.

5.4.3 CONCRETE COMPRESSIVE STRENGTH PREDICTION

This real-world dataset collects 1 030 samples relating eight characteristics of concrete components with the resulting compressive strength (I-Cheng Yeh, 1998). A complete description of all the features can be found in Appendix 5.A.1.

Based on the dataset introductory paper (Yeh, 1998), we assume the relationship between age and compressive strength to be logarithmic, i.e.,

$$h_k^\phi(\text{age}) = \phi_0 \ln(\text{age}) + \phi_1.$$

Given that this experiment is based on real data, we do not have access to ground truth values of ϕ , therefore we can only experiment with the approximate constraining methods and cannot measure $\mathcal{L}_{PD}$. We choose to use 100, 50 and 25 training (and validation) samples, leaving the rest as test set, and set the batch size to 5 samples. For each training size, we generate 20 different dataset splits.

Sample size effect We observe in Figure 5.16 (left) that constraining the PD of the network to be logarithmic helps to reduce the errors made in the final predictions, even more for smaller training sizes. With enough training samples, the gap w.r.t. unconstrained training is closed.

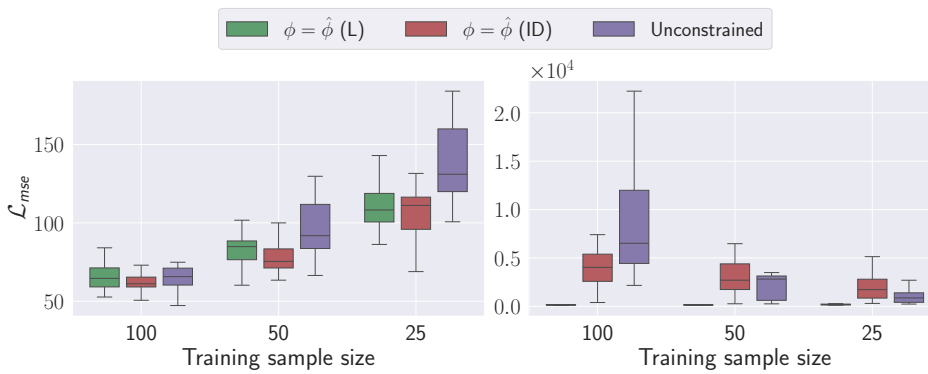


Figure 5.16. Evolution of $\mathcal{L}_{mse}$ w.r.t. the training sample size, on the concrete problem, constraining $PD_N(h^\theta, \text{age})$, in the interpolation setting (left) and extrapolation setting (right). Each boxplot summarizes the results on a test set, over 20 different dataset splits.

Test-time domain shift In this setting, we have removed from training and validation sets samples whose age value was larger than 15 days, which allows to have a shared test set of 706 samples for all extrapolation experiments.

We observe in Figure 5.16 (right) that unconstrained models suffer from shifting outside the age range considered during training. With enough training samples, in-distribution constraining is able to dampen this effect, but it then levels with unconstrained training, and even worsens for smaller training sizes. Linspace constraining, however, vastly outperforms both in-distribution constraining and unconstrained training, for all training sizes.

In Figure 5.17, we provide the mean PD estimations for all methods, both in the interpolation (top) and extrapolation (bottom) settings. In the interpolation setting, we can observe that, with enough training data (100 samples, top left), the PD estimations of all models follow the same logarithmic trend. However, when reducing the number of training samples, we notice that those of unconstrained models tend to diverge and show increased variance. In the extrapolation setting, the situation is radically different. Indeed, both unconstrained models and in-distribution constraining fail to capture the logarithmic shape outside the training domain, which confirms the poor performance that was observed for such methods in Figure 5.16 (right). For this problem, it is clear that the linspace constraint helps to better model the shape of the desired PD, thereby reducing the prediction error, as was observed in Figure 5.16 (right).

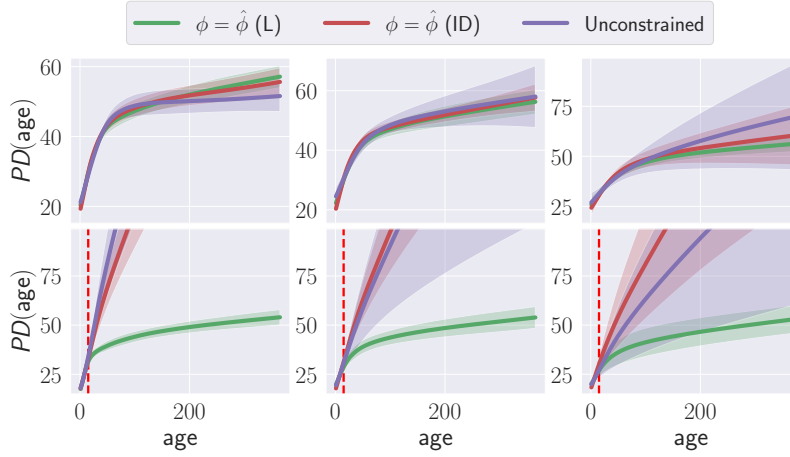


Figure 5.17. Mean PD estimations for $PD_N(h^\theta, \text{age})$ obtained with each method, for the compressive strength problem, in the interpolation (top) and extrapolation settings (bottom). All methods have been trained respectively on 100 samples (left), 50 samples (middle), 25 samples (right). Mean and standard deviations are computed over 20 different dataset splits. The dotted red lines represent the boundaries of the interpolation range.

5.4.4 PHALK DATASET

This dataset relates environmental basin characteristics, alkalinity and pH values from a very large number of inland water sites (Catalan et al., 2025), for a total of 1 017 498 samples. A complete description of all the features can be found in Appendix 5.A.2.

As mentioned in the introductory paper (Batalla et al., 2026), we assumed a logarithmic relationship between pH and alkalinity. Therefore, we decided to train a model to predict pH from the other features, and constrained its partial dependence w.r.t. alkalinity to be logarithmic, i.e.,

$$h_k^\phi(\text{alkalinity}) = \phi_0 \ln(\text{alkalinity}) + \phi_1.$$

As for the previous experiment, we use 100, 50 and 25 training (and validation) samples, leaving the rest as test set, and set the batch size to 5 samples. For each training size, we

generate 20 different dataset splits.

Sample size effect Similarly to the concrete compressive strength prediction problem, we observe in Figure 5.18 (left) that constraining the PD of the model to be logarithmic improves final performance, especially for smaller training sizes.

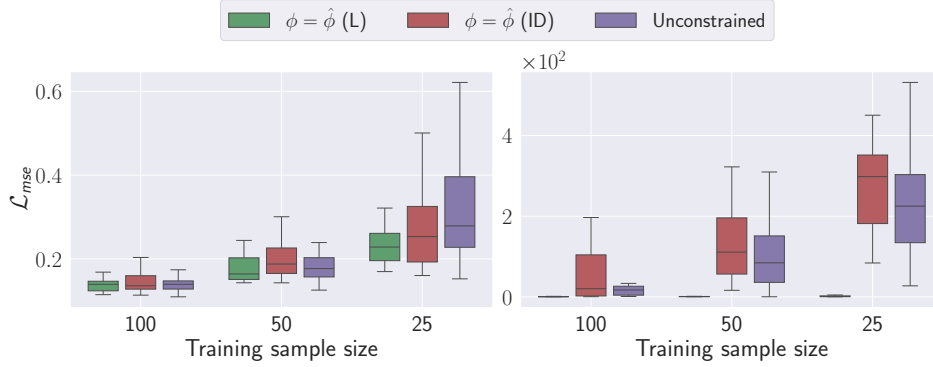


Figure 5.18. Evolution of $\mathcal{L}_{mse}$ w.r.t. the training sample size, on the PHALK dataset, constraining $PD_N(h^\theta, \text{alkalinity})$, in the interpolation setting (left) and extrapolation setting (right). Each boxplot summarizes the results on a test set, over 20 different dataset splits.

Test-time domain shift As typical alkalinity values observed in the dataset lie in $[0, 8000]$, we set a threshold to 250 for the extrapolation setting, i.e., training (and validation) samples only have alkalinity values lower than 250, while test samples have alkalinity values higher than 250.

We observe in Figure 5.18 (right) that, once again, linspace constraining vastly outperforms both in-distribution constraining and unconstrained training. For this problem, in-distribution constraining yields the worst models.

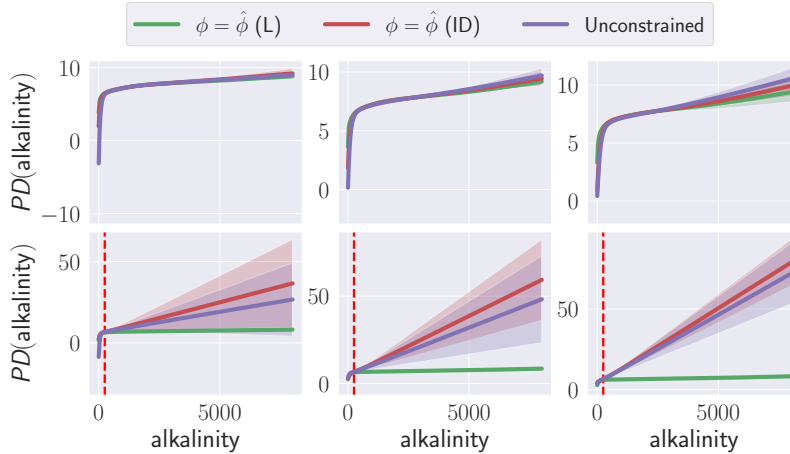


Figure 5.19. Mean PD estimations for $PD_N(h^\theta, \text{alkalinity})$ obtained with each method, on the PHALK dataset, in the interpolation (top) and extrapolation settings (bottom). All methods have been trained respectively on 100 samples (left), 50 samples (middle), 25 samples (right). Mean and standard deviations are computed over 20 different dataset splits. The dotted red lines represent the boundaries of the interpolation range.

In Figure 5.19, we provide the mean PD estimations, both in the interpolation (top) and extrapolation (bottom) settings. In the interpolation setting, we observe that the PD estimations of all models follow the same logarithm-like trend, for all training sizes.

In the extrapolation setting, the situation is quite different. As was the case for the previous experiment, both unconstrained models and in-distribution constraining do not manage to recover the logarithmic structure outside the training domain, which is verified with their respective performance values in Figure 5.18 (right). Linspace constraining, on the contrary, helps to better model the shape of the desired PD, thereby improving final performance.

5.4.5 COMPARISON TO PHYSICS-INFORMED MODELS

In this section, we compare the performance of our constraining methods against two physics-informed machine learning baselines, namely APHYNITY (Yin et al., 2021) and partial-dependence-based hybrid modeling (Claes, Huynh-Thu, et al., 2025, Chapter 3). Although the goal of both approaches is radically different from ours, we deemed interesting to have a look at this specific type of methods in terms of prediction performance but mostly in terms of PD estimation performance, as they are straightforward to implement for the considered nature of knowledge.

APHYNITY Formally, APHYNITY combines a first-principles model, i.e. the prior knowledge $h_k(\mathbf{x}_k; \phi)$, with a ML model denoted $h_a(\mathbf{x}; \theta)$, in an additive manner, with learnable ϕ and θ . The balance between h_k and h_a is controlled by minimizing the contribution of h_a while trying to minimize the prediction errors of the entire model $h = h_k + h_a$, through the optimization of some regularization coefficient $\lambda \in \mathbb{R}$.

Partial-dependence-based hybrid modeling Similarly to APHYNITY, this hybrid method also additively combines the first-principles model h_k with the ML model h_a . However, the optimization of both models is no longer performed jointly. Indeed, $h_k(\mathbf{x}_k; \phi)$ is first optimized by iteratively fitting the latter on PD estimations of an ML model. Then, $h_a(\mathbf{x}; \theta)$ is trained once on the residual $y - h_k(\mathbf{x}_k; \phi)$.

For all experiments, we fix the architecture of the neural network to three layers of increasing hidden size (32, 64 and 128), with a fixed training batch size of 16 samples (resp. 4 trajectories for dynamical systems and 5 for the real-world problems). We set the number of epochs $E = 1000$. For APHYNITY, we optimize the joint learning rate in $\{5 \times 10^{-2}, 5 \times 10^{-3}, 5 \times 10^{-4}\}$ and the coefficient λ in $\{1, 10, 20\}$, selecting the values that minimize $\mathcal{L}_{mse}$ on the validation set. We refer to partial-dependence-based hybrid modeling as “PD hybrid” in the following results.

FRIEDMAN PROBLEM

Let us recall the problem definition (Friedman et al., 1983):

$$y = \psi_0 \sin(\psi_1 x_0 x_1) + \psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4 + \sum_{j=5}^9 0x_j + \varepsilon,$$

where $x_j \sim \mathcal{U}(0, 1)$, $j = 0, \dots, 9$, $\psi = [10, \pi, 20, 0.5, 10, 5]$ and $\varepsilon \sim \mathcal{N}(0, 1)$.

For hybrid methods, we assume prior knowledge similar to our method and specifically focus on a single setting for this problem, i.e.

$$h_k(x_0, x_1; \phi) = \phi_0 [\phi_2 \sin(\phi_3 x_0 x_1)] + \phi_1,$$

and the final model writes $h(\mathbf{x}) = h_k(x_0, x_1; \phi) + h_a(\mathbf{x}; \theta)$.

Sample size effect We observe in Figure 5.20 that hybrid methods perform the best for both measures of performance. This can actually be explained by the structure of this problem and by the inductive bias of hybrid additive methods. Indeed, given that this problem is purely additive with independent input features, it is therefore easy for such hybrid methods to converge towards faithful estimations of the prior knowledge. Note that this particular problem structure is also beneficial for constrained models with approximated priors, as the obtained priors are of better quality. Still, our method remains less restrictive as it does not assume model additivity but rather relies on a single model.

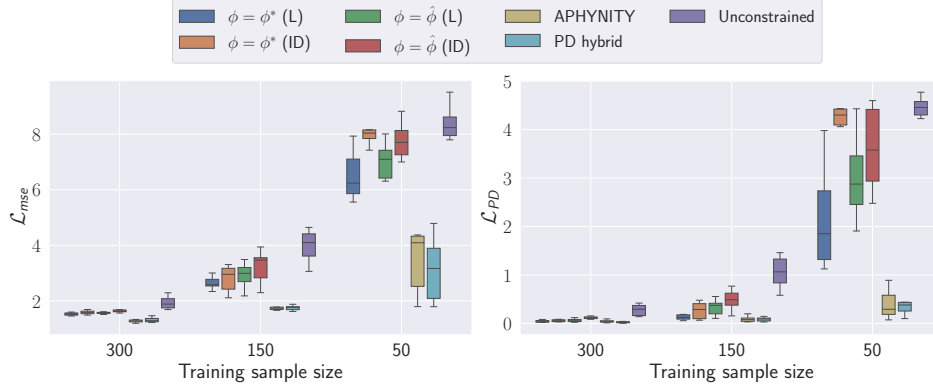


Figure 5.20. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_0, x_1)$. Each boxplot summarizes the results over 10 different training initializations.

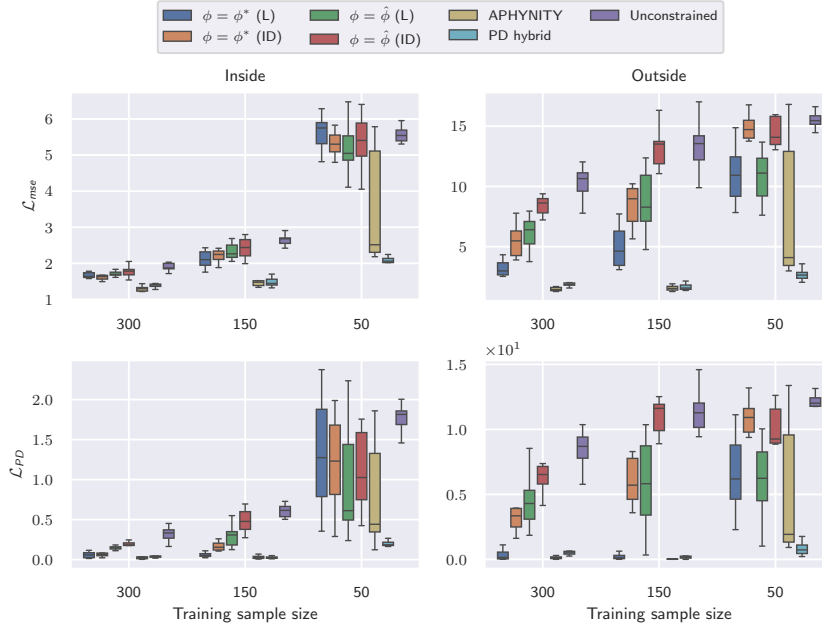


Figure 5.21. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_0, x_1)$. *Inside* metrics are computed on samples for which $x_0 \leq 0.75$ (left plots), while *Outside* metrics are computed on samples where $x_0 > 0.75$ (right plots). Each boxplot summarizes the results on a test set, over 10 different training initializations.

Test-time domain shift In Figure 5.21, we observe that hybrid methods are quite robust to test-time domain shift when training on 300 and 150 samples. We nevertheless notice that, when training on scarcer data (50 samples), the gap in performance on both measures is tightened between APHYNITY and linspace constraining, especially for $\mathcal{L}_{PD}$. This could be explained by the fact that, due to the limited training domain, the quality of h_k degrades and the statistical model h_a then tries to compensate the misfit, which might lead to a worse PD. With linspace constraining, although the approximated prior also deteriorates, the constraint allows to regularize the shape of the PD outside the originally considered training domain. Still, we observe that PD-based hybrid modeling is not impaired by the extrapolation setting, which suggests that, for this problem, the iterative training procedure of h_k on PD (residual) estimations of h_a yields better physical parameters than those obtained with APHYNITY.

PRODUCT PROBLEM

Let us recall the problem definition:

$$y = \psi_0 x_0^3 \cos(\psi_1 x_0) \frac{(x_1 - 1)^2}{2} + \sum_{j=2}^9 0x_j + \varepsilon,$$

where features are correlated with zero mean, variances and covariances equal to 0.5 and 0.25, respectively, $\psi = [15, \pi]$, and $\varepsilon \sim \mathcal{N}(0, 1)$.

For hybrid methods, we assume prior knowledge through

$$h_k(x_0; \phi) = \phi_0 [\phi_2 x_0^3 \cos(\phi_3 x_0)] + \phi_1,$$

and the final model writes $h(\mathbf{x}) = h_k(x_0; \phi) + h_a(\mathbf{x}; \theta)$.

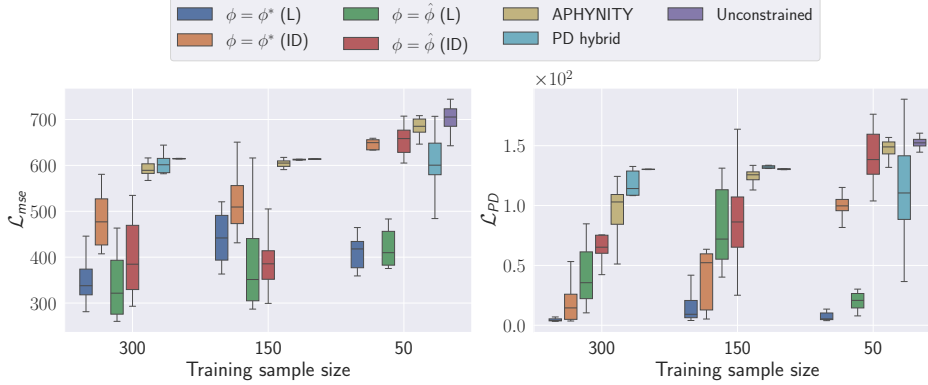


Figure 5.22. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the product problem, constraining $PD_N(h^\theta, x_0)$. Each boxplot summarizes the results over 10 different training initializations.

Sample size effect We observe in Figure 5.22 that hybrid methods perform only slightly better than unconstrained models, for all training sizes. This can be explained by the fact that such methods explicitly model additivity, which is a wrong assumption for this problem. We can notice that with sufficient data (300 samples), their PD estimation becomes slightly better than what would be obtained with an unconstrained neural network. On the contrary, our constraining method does not suffer from this particular problem structure, given that it does not encode additivity but simply relies on a single ML model. Although in-distribution constraining degrades with scarce data (50 samples) and performs roughly on par with hybrid models, linspace constraining vastly outperforms the other methods, for both measures.

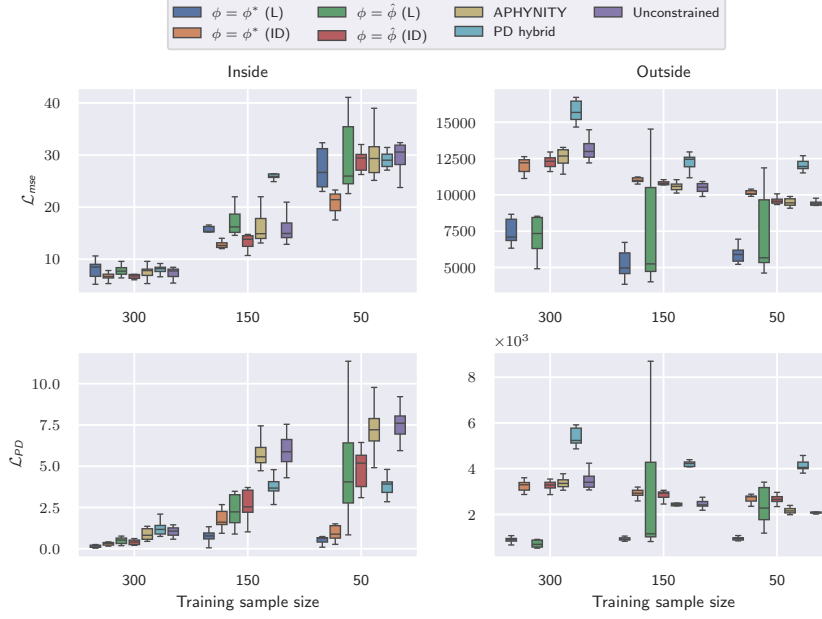


Figure 5.23. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the product problem, constraining $PD_N(h^\theta, x_0)$. *Inside* metrics are computed on samples for which $x_0 \in [-1.25, 1.25]$ (left plots), while *Outside* metrics are computed on samples outside that range (right plots). Each boxplot summarizes the results on the test set, over 10 different training initializations.

Test-time domain shift In Figure 5.23, we notice that even though the gap is closed inside the interpolation domain, their PD estimation still remains worse than that of constrained models, and equivalent to that of unconstrained networks. For samples outside the interpolation range, we can observe that linspace constraining also outperforms hybrid models.

CONCRETE COMPRESSIVE STRENGTH PREDICTION

As a reminder, the dataset is composed of samples relating characteristics of concrete components with the associated compressive strength. For hybrid methods, we assume similar logarithmic knowledge, i.e.

$$h_k(\text{age}; \phi) = \phi_0 \ln(\text{age}) + \phi_1,$$

and the final model writes $h(\mathbf{x}) = h_k(\text{age}; \phi) + h_a(\mathbf{x})$.

Sample size effect We observe in Figure 5.24 (left) that, as constrained models, hybrid methods reach better performance than unconstrained networks, for all training sizes. The gap with constrained methods is nonexistent, and we note that the performance of APHYNITY and linspace constraining are closely aligned.

Test-time domain shift We observe in Figure 5.24 (right) that APHYNITY falls short when facing out-of-distribution samples, performing roughly identically to in-distribution constraining and unconstrained models, meaning that, even though it is encoded in the model itself, it does not seem to be able to capture the logarithmic signal in the resulting PD, compared to linspace constraining. PD hybrid modeling performs much better than APHYNITY, but still slightly worse than linspace constraining with sufficient training

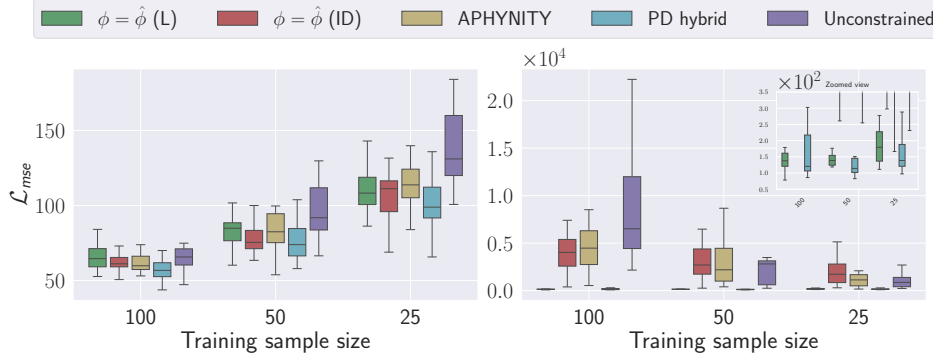


Figure 5.24. Evolution of $\mathcal{L}_{mse}$ w.r.t. the training sample size, on the concrete problem, constraining $PD_N(h^\theta, \text{age})$, in the interpolation setting (left) and extrapolation setting (right). Each boxplot summarizes the results on a test set, over 20 different dataset splits.

samples, which suggests that the logarithmic effect of age could not be entirely additive, otherwise the gap would be tighter.

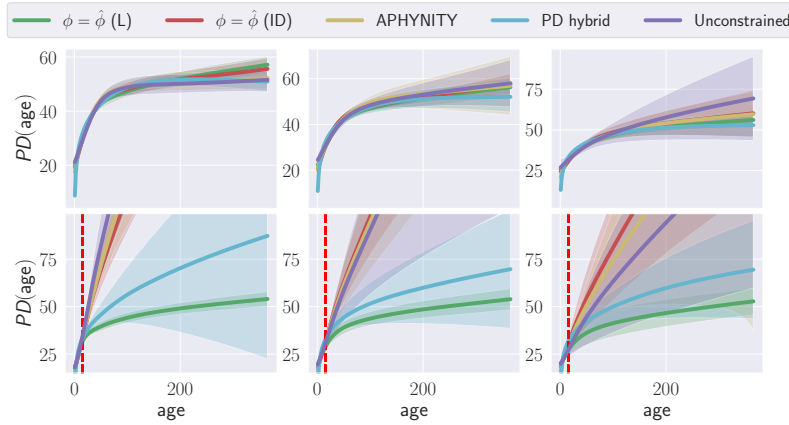


Figure 5.25. Mean PD estimations for $PD_N(h^\theta, \text{age})$ obtained with each method, for the compressive strength problem, in the interpolation (top) and extrapolation settings (bottom). All methods have been trained respectively on 100 samples (left), 50 samples (middle), 25 samples (right). Mean and standard deviations are computed over 20 different dataset splits. The dotted red lines represent the boundaries of the interpolation range.

We represent in Figure 5.25 the mean PD estimations, for all methods and training sizes. In particular, we can notice that in the interpolation setting (top), the PD estimations all follow a logarithmic shape. However, in the extrapolation setting (bottom), the PD estimations of APHYNITY completely diverge and follow the same trend as those of unconstrained models, showing that even though APHYNITY encodes the logarithmic relationship by construction, it is insufficient to result in a reliable PD. On the contrary, PD hybrid modeling suffers less than APHYNITY, as its PD estimation looks close to that of linspace constraining.

PHALK DATASET

As a reminder, the dataset contains samples of basin characteristics, pH and alkalinity values, and we aim to predict pH values. For hybrid methods, we assume logarithmic

knowledge, i.e.,

$$h_k(\text{alkalinity}; \phi) = \phi_0 \ln(\text{alkalinity}) + \phi_1,$$

and the final model writes $h(\mathbf{x}) = h_k(\text{alkalinity}; \phi) + h_a(\mathbf{x})$.

Sample size effect We observe in Figure 5.26 (left) that hybrid methods reach better performance than unconstrained networks, for all training sizes.

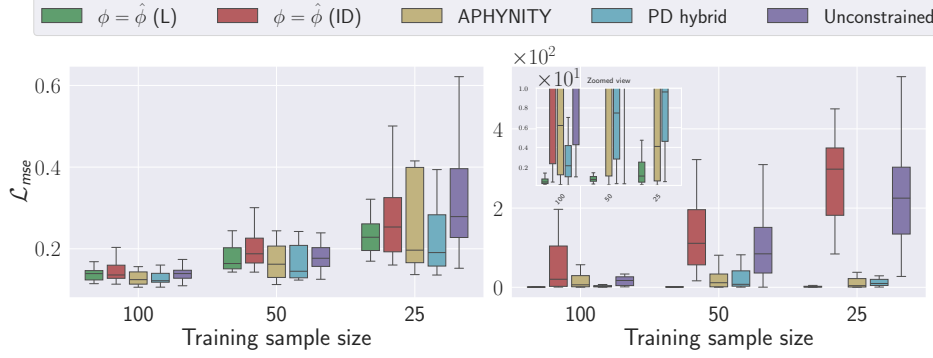


Figure 5.26. Evolution of $\mathcal{L}_{mse}$ w.r.t. the training sample size, on the PHALK dataset, constraining $PD_N(h^\theta, \text{alkalinity})$, in the interpolation setting (left) and extrapolation setting (right). Each boxplot summarizes the results on a test set, over 20 different dataset splits.

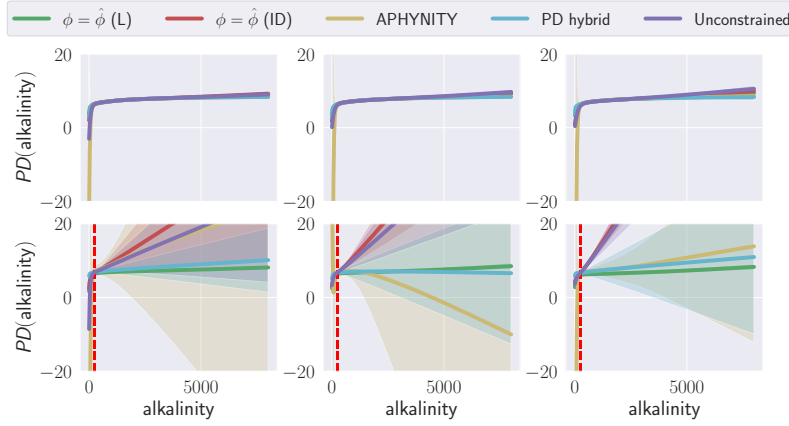


Figure 5.27. Mean PD estimations for $PD_N(h^\theta, \text{alkalinity})$ obtained with each method, for the PHALK dataset, in the interpolation setting (top) and extrapolation settings (bottom). All methods have been trained respectively on 100 samples (left), 50 samples (middle), 25 samples (right). Mean and standard deviations are computed over 20 different dataset splits. The dotted red lines represent the boundaries of the interpolation range.

Test-time domain shift We observe in Figure 5.26 (right) that hybrid methods reach worse performance with out-of-distribution samples, yet performing better than in-distribution constraining and unconstrained models. For this problem, we observe that linspace constraining vastly outperforms all the other methods.

We represent in Figure 5.27 the mean PD estimations, for all methods and training sizes. We notice that in the interpolation setting (top), the PD estimations all follow a logarithmic trend. However, in the extrapolation setting (bottom), the PD estimations

of APHYNITY show stability problems, but generally do not look logarithmic. On the contrary, PD hybrid modeling suffers less than APHYNITY, as its PD estimation looks close to that of linspace constraining.

5.4.6 DYNAMICAL SYSTEMS FORECASTING

Our algorithm can be straightforwardly extended to the problem of dynamical systems forecasting. In this problem, we are interested in predicting the evolution of a system driven by some dynamics f

$$\dot{\mathbf{x}}_t := \frac{d\mathbf{x}_t}{dt} = f(\mathbf{x}_t), \quad (5.10)$$

where $\mathbf{x}_t \in \mathbb{R}^d$, and $f(\mathbf{x}_t)$ shares the same dimension as $\mathbf{x}_t$. Similarly, we assume knowledge about the dynamics function in the form of a PD function, and re-use previous model definitions of h^θ and h_k^ϕ .

We assume access to a finite set LS of N observed state trajectories over a given time interval $[0, T]$, where T is the time horizon, i.e. $LS = \{\mathbf{x}_0^{(i)}, \dots, \mathbf{x}_T^{(i)}\}_{i=1}^N$. We are still interested in solving Equation (5.7), i.e.

$$\theta^*, \phi^* = \arg \min_{\theta, \phi} \mathcal{L}_\lambda(h^\theta, h_k^\phi, f; LS),$$

with

$$\mathcal{L}_\lambda(h^\theta, h_k^\phi, f; LS) = (1 - \lambda)\mathcal{L}_{mse}(h^\theta, f; LS) + \lambda\mathcal{L}_{PD}(h^\theta, h_k^\phi; LS),$$

for which

$$PD_N(h^\theta, \mathbf{x}_k) = \frac{1}{T} \frac{1}{N} \sum_{t=1}^T \sum_{i=1}^N h^\theta(\mathbf{x}_k, \mathbf{x}_{t,-k}^{(i)}), \quad (5.11)$$

$$\mathcal{L}_{mse}(h^\theta, f; LS) = \frac{1}{T} \frac{1}{N} \sum_{t=1}^T \sum_{i=1}^N \|\hat{\mathbf{x}}_t^{(i)} - \mathbf{x}_t^{(i)}\|^2, \quad (5.12)$$

$$\mathcal{L}_{PD}(h^\theta, h_k^\phi; LS, LS_{PD}) = \frac{1}{N_{PD}} \sum_{j=1}^{N_{PD}} (PD_N(h^\theta, \mathbf{x}_k^{(j)}) - h_k^\phi(\mathbf{x}_k^{(j)}))^2, \quad (5.13)$$

where $\mathbf{x}_k$ is any value of the features of interest, $\hat{\mathbf{x}}_t^{(i)}$ is the solution of integrating h^θ with initial condition $\mathbf{x}_0^{(i)}$ up to time t , and $LS_{PD} = \{\mathbf{x}_k^{(j)}\}_{j=1}^{N_{PD}}$ is a set of arbitrary values for the subset of features of interest.

As was the case before, we need to initialize the process, hence we arbitrarily fit h_k^ϕ for E epochs on the raw trajectories, yielding

$$\phi^{(0)} = \arg \min_{\phi} \mathcal{L}_{prior}(h_k^\phi, f; LS),$$

with

$$\mathcal{L}_{prior}(h_k^\phi, f; LS) = \frac{1}{T} \frac{1}{N} \sum_{t=1}^T \sum_{i=1}^N \|\hat{\mathbf{x}}_{t,k}^{(i)} - \mathbf{x}_{t,k}^{(i)}\|^2, \quad (5.14)$$

where $\hat{\mathbf{x}}_{t,k}^{(i)}$ is the solution of integrating h_k^ϕ with initial condition $\mathbf{x}_{0,k}^{(i)}$ up to time t .

DAMPED PENDULUM SYSTEM

Let us consider the following dynamical system:

$$\begin{aligned} \frac{d^2 x_0}{dt^2} + \psi_0 \sin x_0 + \psi_1 \frac{dx_0}{dt} &= 0, \\ \Leftrightarrow \begin{cases} \dot{x}_0 = \frac{dx_0}{dt} \\ \ddot{x}_0 = -\psi_0 \sin x_0 - \psi_1 \frac{dx_0}{dt} \end{cases} \end{aligned}$$

with $\mathbf{x} = (x_0, \frac{dx_0}{dt})$ the state, whose initial values are respectively sampled from $\mathcal{U}(-\pi, \pi)$ and $\mathcal{U}(0, \frac{1}{1000})$, and $\psi = [(\frac{\pi}{6})^2, 0.2]$, with the following PD function:

$$h_k^\phi(x_0) = \phi_0[-\phi_2 \sin x_0] + \phi_1.$$

For this problem, even though prior knowledge is only defined for $\ddot{x}_0$ through the PD, we still need a dynamical model for x_0 to track its evolution over time as it serves as the argument for the PD. Given that $x_1 = \frac{dx_0}{dt}$, we therefore redefine $h_k^\phi : (x_0, x_1) \rightarrow (x_1, \phi_0[-\phi_2 \sin x_0] + \phi_1)$.

A potential ground truth parameter vector is $\phi^* = [1, C, \psi_0]$, where:

$$C = \mathbb{E}_{\frac{dx_0}{dt}} \left[-\psi_1 \frac{dx_0}{dt} \right].$$

We generate training and validation sets of successively 50, 25 and 10 trajectories each. Each trajectory spans 81 evenly spaced timesteps (including the initial state $\mathbf{x}_0$), over a time interval of $[0, 20]$ s, integrated through the Neural ODE approach (R. T. Q. Chen et al., 2018), using dopri5 as solver. All methods are then evaluated on a fixed test set of 200 trajectories, and we set the batch size to 4 trajectories.

Sample size effect As can be seen in Figure 5.28, while constraining seems to have moderate impact on the predictive performance for a learning sample of 50 trajectories, it nevertheless helps to predict more accurate trajectories for smaller training sample sizes, especially for linspace constraining methods (blue and green boxes). This positive impact is reflected on the quality of the learned PD function, for all constraining approaches, although more deeply for linspace constraining. Furthermore, linspace constraining on $h_k^{\phi^*}$ reaches similar predictive performance compared to unconstrained training but with only 25 training trajectories, compared to 50 for the latter. The same tendency can be observed for linspace constraining on an approximated prior, though it slightly suffers from the variance of the learned prior function. Finally, in-distribution constraining does not really improve predictive performance, but helps to learn a better PD function for smaller training sizes.

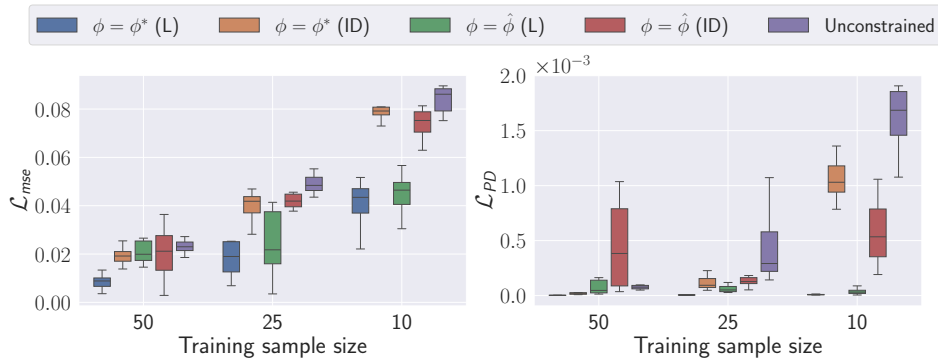


Figure 5.28. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the damped pendulum problem, constraining $PD_N(h^\theta, x_0)$. Each boxplot summarizes the results on the test set, over 10 different training initializations.

Test-time domain shift To study test-time shift, initial states whose angle $x_0 \notin [-\frac{3\pi}{4}, \frac{3\pi}{4}]$ are rejected from the training and validation sets. We observe in Figure 5.29 that predictive performance inside the interpolation range is now much lower for all

training sizes and all methods, as a result of reducing the size of the input domain for the initial condition. As was the case previously, 50 training trajectories seem to be enough for unconstrained training to be as accurate *inside* as constraining approaches, although producing slightly worse PD estimations. As the number of trajectories decreases, the gap between linspace approaches and unconstrained training becomes larger, and the former even perform on par with the latter with only 10 trajectories, compared to 25. Similarly to the previous experiment, in-distribution constraining does not really improve predictive performance (neither *inside* nor *outside* the interpolation range), but helps to learn better *inside* PD estimates. However, it is striking that linspace constraining outperforms unconstrained training when starting from out-of-distribution initial states, both in terms of predictive performance and PD estimates.

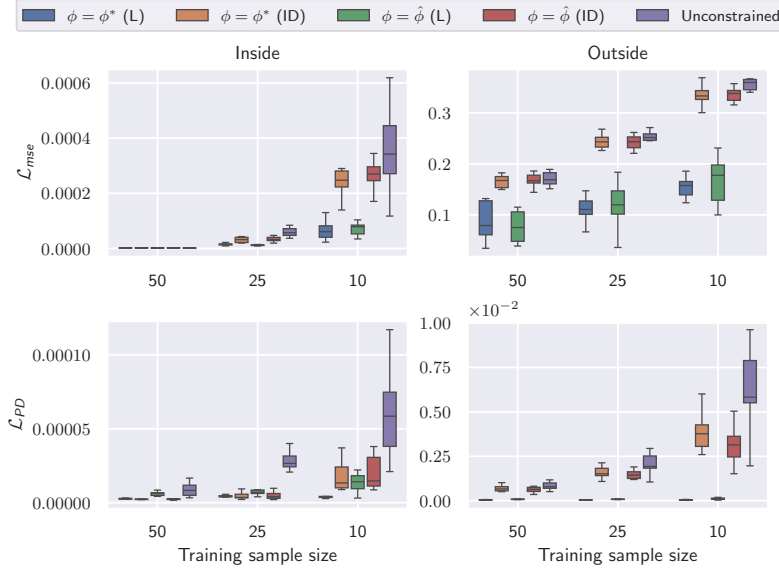


Figure 5.29. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the damped pendulum problem, constraining $PD_N(h^\theta, x_0)$. *Inside* metrics are computed on samples for which the initial $x_0 \in [-\frac{3\pi}{4}, \frac{3\pi}{4}]$ (left plots), while *Outside* metrics are computed on samples outside that range (right plots). Each boxplot summarizes the results on the test set, over 10 different training initializations.

5.4.7 CONSTRAINING WITH MIS-SPECIFIED KNOWLEDGE

In this section, we study the performance of our constraining methods in the presence of mis-specified prior knowledge, thereby assuming that the expert poorly models the true partial dependence function. For that purpose, we rely on models whose hypothesis spaces do not include the true PD model but respect similar properties.

For both experiments, we fix the architecture of the neural network to three layers of increasing hidden size (32, 64 and 128), with a fixed training batch size of 16 samples for the Friedman problem, and 5 for the concrete compressive strength problem. We optimize the learning rate τ_2 in $\{5 \times 10^{-2}, 5 \times 10^{-3}, 5 \times 10^{-4}\}$ and the coefficient λ in $\{0.25, 0.5, 0.75\}$, selecting the values that minimize $\mathcal{L}_{mse}$ on the validation set.

FRIEDMAN PROBLEM

Let us recall the problem definition (Friedman et al., 1983):

$$y = \psi_0 \sin(\psi_1 x_0 x_1) + \psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4 + \sum_{j=5}^9 0x_j + \varepsilon,$$

where $x_j \sim \mathcal{U}(0, 1)$, $j = 0, \dots, 9$, $\psi = [10, \pi, 20, 0.5, 10, 5]$ and $\varepsilon \sim \mathcal{N}(0, 1)$.

In this setting, we only assume the prior knowledge to be a periodic function

$$h_k^\phi(x_0, x_1; \phi, \phi_{\sin}, \phi_{\cos}) = \phi_0 \left[\sum_{l=1}^L \phi_{\sin}^{(l)} \sin(2\pi l x_0 x_1) + \phi_{\cos}^{(l)} \cos(2\pi l x_0 x_1) \right] + \phi_1 \quad (5.15)$$

through a Fourier series with period 1, where $\phi_{\sin} \in \mathbb{R}^L$, $\phi_{\cos} \in \mathbb{R}^L$. We denote linspace constraining with this imperfect setting as $\phi = \hat{\phi}$ (L - Imperf.), and in-distribution constraining as $\phi = \hat{\phi}$ (ID - Imperf.).

Let us remind that the true PD function should be

$$h_k^{\phi^*}(x_0, x_1) = 1 \cdot [10 \sin(\pi x_0 x_1)] + C_1,$$

where

$$C_1 = \mathbb{E}_{\mathbf{x}_{-k}} [\psi_2 (x_2 - \psi_3)^2 + \psi_4 x_3 + \psi_5 x_4].$$

However, given that $h_k^\phi(x_0, x_1)$ here solely relies on basis terms with frequencies that are integer multiples of 2π , it is therefore impossible to recover exactly the true PD function with h_k^ϕ , hence the knowledge is imperfect by construction, but it is still able to provide a decent representation. For this experiment, we set $L = 3$.

Sample size effect We observe in Figure 5.30 that decreasing the learning sample size degrades both measures of performance. Nevertheless, constraining the PD of the model with such imperfect knowledge still improves both measures compared to unconstrained training, for all training sizes.

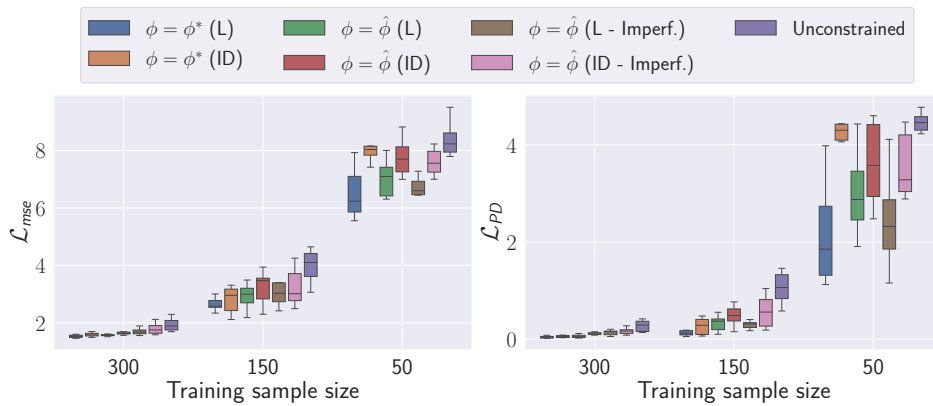


Figure 5.30. Evolution of $\mathcal{L}_{mse}$ (left) and $\mathcal{L}_{PD}$ (right) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_0, x_1)$. Each boxplot summarizes the results on the test set, over 10 different training initializations.

Even with mis-specified prior knowledge, linspace constraining (brown boxes) performs on par with in-distribution constraining on true PD samples (orange boxes), and even

outperforms the latter when data becomes very scarce (50 samples). This can be verified by looking at Figure 5.31, which highlights the difference in the PD estimates. In general, constraining with imperfect prior knowledge (brown and pink boxes) performs as well as constraining on the true PD with approximated parameters (green and red boxes), but we notice that the former actually outperforms the latter for 50 samples, which is due to the fact that it does not need to approximate the frequency parameter of the sine function, making it less prone to overfitting.

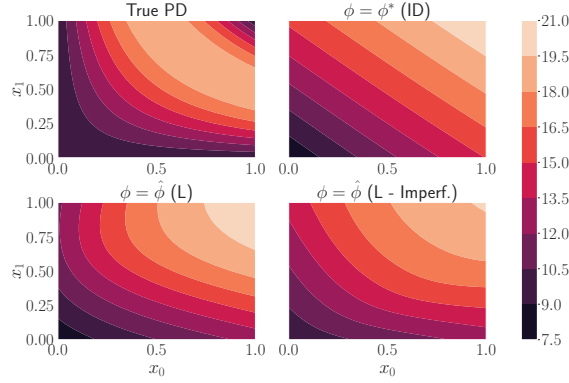


Figure 5.31. Examples of PD estimation for $PD_N(h^\theta, x_0, x_1)$. The figure displays the true PD (upper left), the PD after in-distribution constraining on the ground truth PD (upper right), the PD after linspace constraining on the approximated prior (bottom left), and the PD after linspace constraining with imperfect prior knowledge (bottom right). All have been trained on 50 samples, in the interpolation setting.

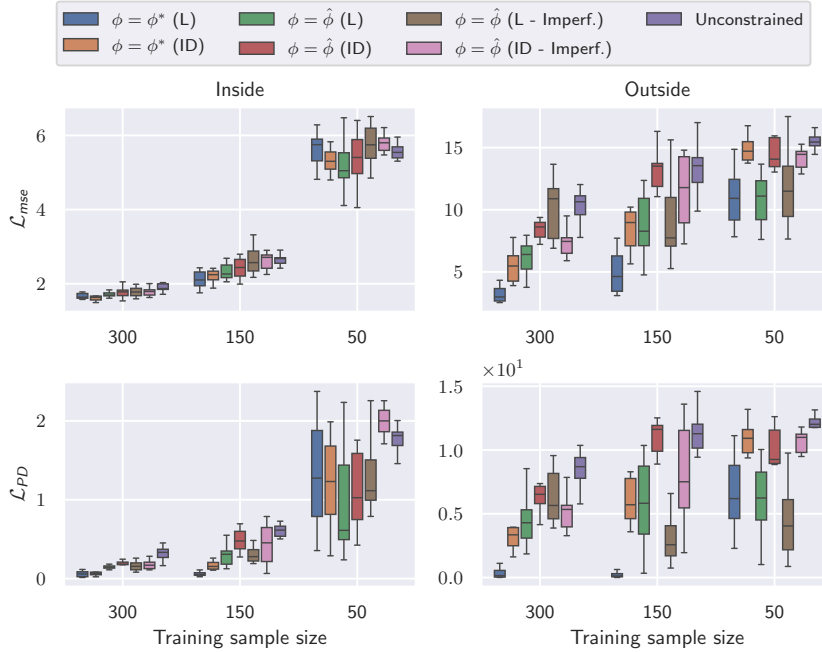


Figure 5.32. Evolution of $\mathcal{L}_{mse}$ (top plots) and $\mathcal{L}_{PD}$ (bottom plots) w.r.t. the training sample size, on the Friedman problem, constraining $PD_N(h^\theta, x_0, x_1)$. *Inside* metrics are computed on samples for which $x_0 \leq 0.75$ (left plots), while *Outside* metrics are computed on samples where $x_0 > 0.75$ (right plots). Each boxplot summarizes the results on the test set, over 10 different training initializations.

Test-time domain shift We measure performance under test-time domain shift in identical conditions, i.e. samples for which $x_0 > 0.75$ are rejected from the training and validation sets.

We can observe in Figure 5.32 that, for both metrics, PD steering helps to mitigate the impact of test-time domain shift, and much more when constraining on evenly-spaced points. As for the interpolation setting, we can notice that our method is robust to imperfect knowledge in the sense of Equation (5.15).

CONCRETE COMPRESSIVE STRENGTH PREDICTION

As a reminder, the dataset is composed of samples relating characteristics of concrete components with the associated compressive strength. Instead of assuming the relationship between age and compressive strength to be logarithmic, we model the latter as a square root

$$h_k(\text{age}; \phi) = \phi_0 \sqrt{\text{age}} + \phi_1.$$

The motivation behind this choice is to assume that prior knowledge is limited to a strictly increasing concave function, hence the expert wrongly modeled the PD with a square root function.

Sample size effect We observe in Figure 5.33 (left) that when constraining with imperfect knowledge, the performance of the resulting models is no better than that of the unconstrained models, which was not the case when assuming a logarithmic PD. This behavior actually makes sense, since we are trying to constrain the PD of the neural network to follow an incorrect shape.

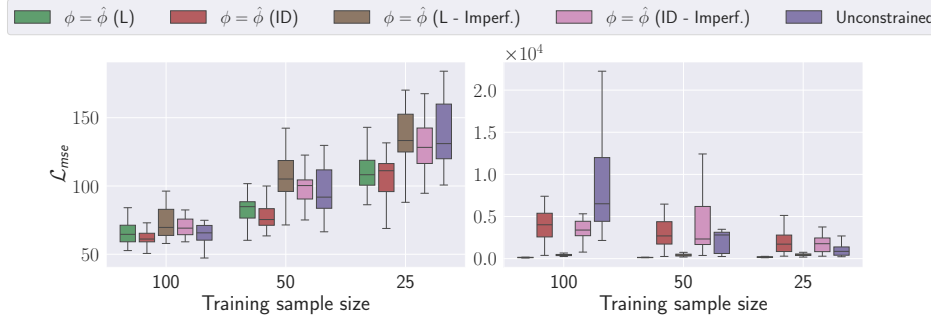


Figure 5.33. Evolution of $\mathcal{L}_{mse}$ w.r.t. the training sample size, on the concrete problem, constraining $PD_N(h^\theta, \text{age})$, in the interpolation setting (left) and extrapolation setting (right). Each boxplot summarizes the results on a test set, over 20 different dataset splits.

Contrary to the previous problem where periodicity was the only assumption, the source of mis-specification in this case is tougher as there is no way for a square root function to yield a reasonable approximation of a logarithmic function.

Test-time domain shift Identically to the original extrapolation setting, we have removed from training and validation sets samples whose age value was larger than 15 days. We observe in Figure 5.33 (right) that, in this case, the performance of models constrained with imperfect prior knowledge is actually very close to that of models constrained with the logarithmic PD. As was the case, linspace constraining outperforms both in-distribution constraining and unconstrained training, for all training sizes, and reaches performance close to that obtained in the interpolation setting, even though prior knowledge is fundamentally mis-specified.

In Figure 5.34, we provide the mean PD estimations for all methods, both in the interpolation (top) and extrapolation (bottom) settings. In the interpolation setting, we can observe that the PD estimations of models constrained with imperfect knowledge indeed follow a square root trend, as opposed to the logarithmic trend observed for the other models. Interestingly, when data becomes very scarce (25 samples), we notice that those of unconstrained models tend to approach the square root trend.

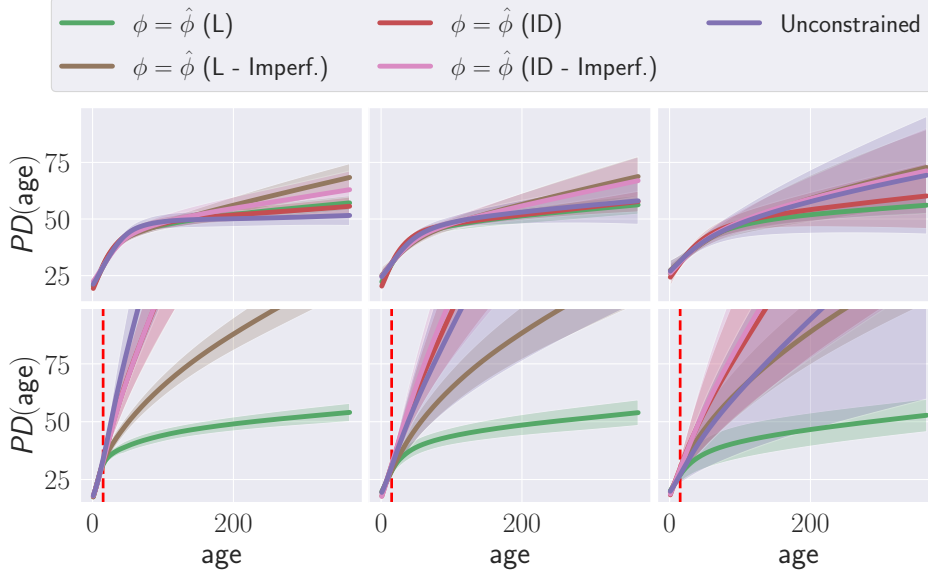


Figure 5.34. Mean PD estimations for $PD_N(h^\theta, \text{age})$ obtained with each method, for the compressive strength problem, in the interpolation (top) and extrapolation settings (bottom). All methods have been trained respectively on 100 samples (left), 50 samples (middle), 25 samples (right). Mean and standard deviations are computed over 20 different dataset splits. The dotted red lines represent the boundaries of the interpolation range.

In the extrapolation setting, the situation is different. Indeed, unconstrained models and both in-distribution constraining (red and pink curves) fail to capture the logarithmic shape outside the training domain, which confirms the poor performance that was observed in Figure 5.33 (right). Even more, we notice that the PD estimations of both in-distribution constraining are very close, which makes sense given the resemblance of the logarithm and square root functions for small inputs. Despite imperfect prior knowledge, linspace constraining helps to approach a shape reasonably close to the desired PD, thereby reducing the prediction error, as was observed in Figure 5.33 (right). With very few data samples (25), we can observe that the PD of unconstrained models approaches that of linspace-constrained models (brown curve), although it still seems to diverge for larger values of age, which explains the gap in prediction error.

5.5 CONCLUSION

We introduce a new approach in explanation-guided learning, based on PD functions, to constrain the training of neural networks. We compare different constraining approaches, depending on the level of available knowledge. On several regression problems, including dynamical systems forecasting and real-world datasets, we empirically show that our constraining methods help to learn models that produce more reliable PD estimates, but also models with stronger predictive performance. In particular, linspace constraining approaches generally perform much better than both

in-distribution training and unconstrained training. Finally, we show that linspace constraining yields robust neural networks when observing test-time domain shift, even with very scarce training data or with mis-specified prior knowledge. We also show that compared to physics-informed hybrid models, our constrained models generally perform better and provide more faithful PD estimates, especially in out-of-distribution settings, as they are more generic and do not explicitly encode additivity.

As shown on the product problem, with correlated input features, we think that our method could be a way to improve the faithfulness of the resulting PDP of the model, in particular when using linspace constraints, as this will force the model to use the features involved in the PDP ($\mathbf{x}_k$) instead of their correlated counterparts.

Our constraining method suffers from larger computing times compared to unconstrained neural networks, due to the estimation of $PD_N(h^\theta, \mathbf{x}_k)$ which in principle requires the evaluation of the model on all N possible pairs $(\mathbf{x}_k, \mathbf{x}_{-k}^{(i)})_{i=1}^N$. It is nevertheless straightforward to reduce the number of data points by randomly sampling $n < N$ points from the original dataset, which would potentially come at the cost of worse partial dependence estimations, but it still remains an unbiased estimator and would reduce training time. Moreover, given that we target mostly low-data regimes, computing times is not expected to be a serious limitation.

As future work, we are interested in testing other methods to learn the initial prior function when $\phi = \hat{\phi}$. Indeed, we have currently only implemented this learning process through a complete fit of $h_k^{\hat{\phi}}$ on the raw output but it might not always be an optimal choice, and the approximation of the PD function is crucial as it would most likely degrade model performance if fitted poorly. Another interesting problem would be to assume no explicit expert knowledge about the shape of the PD function, but rather to rely on an approximation through some fitted model.

Furthermore, our method can in principle be adapted to any other functional interpretation method, e.g., individual conditional expectation (ICE) measures. Therefore, we would like to investigate the impact of constraining with such measures on both predictive performance and explanation quality, which could help in problems with heterogeneous effects that the PD would cancel.

Finally, beyond neural networks, it would be also interesting to adapt the framework to other popular regression methods such as boosting algorithms.

REFERENCES

- Adadi, Amina and Mohammed Berrada (2018). “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)”. In *IEEE access* 6, pp. 52138–52160.
- Amos, Brandon, Lei Xu, and J Zico Kolter (2017). “Input convex neural networks”. In *International conference on machine learning*. PMLR, pp. 146–155.
- Apley, Daniel W and Jingyu Zhu (2020). “Visualizing the effects of predictor variables in black box supervised learning models”. In *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.4, pp. 1059–1086.
- Batalla, Meritxell, Jordi Martínez-Artero, and Jordi Catalan (2026). “Global basin-scale mapping of pH and alkalinity in inland waters”. In *Scientific Data*.
- Caruana, Rich et al. (2015). “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.
- Catalan, J., M. Batalla, and J. Martínez-Artero (2025). “PHALK - A global dataset of estimated pH and alkalinity in inland waters at basin-scale”. Zenodo. DOI: <https://doi.org/10.5281/zenodo.15591825>.
- Chattopadhyay, Aditya et al. (2018). “Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks”. In *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, pp. 839–847.
- Chen, Asic et al. (2023). “Structured neural networks for density estimation and causal inference”. In *Advances in Neural Information Processing Systems* 36, pp. 66438–66450.
- Chen, Ricky T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In *Advances in Neural Information Processing Systems*.
- Claes, Yann, Pierre Geurts, and Vân Anh Huynh-Thu (2026). “Steering Neural Network Training through Interpretable Constraints Based on Partial Dependence”. In *arXiv preprint arXiv:2607.08641*. Submitted to TMLR.
- Claes, Yann, Vân Anh Huynh-Thu, and Pierre Geurts (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Friedman, Jerome H (2001). “Greedy function approximation: a gradient boosting machine”. In *Annals of statistics*, pp. 1189–1232.
- Friedman, Jerome H, Eric Grosse, and Werner Stuetzle (1983). “Multidimensional additive spline approximation”. In *SIAM Journal on Scientific and Statistical Computing* 4.2, pp. 291–301.
- Gao, Yuyang, Siyi Gu, et al. (2024). “Going beyond xai: A systematic survey for explanation-guided learning”. In *ACM Computing Surveys* 56.7, pp. 1–39.
- Gao, Yuyang, Tong Steven Sun, Guangji Bai, et al. (2022). “Res: A robust framework for guiding visual explanation”. In *proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 432–442.
- Gao, Yuyang, Tong Steven Sun, Liang Zhao, et al. (2022). “Aligning eyes between humans and deep neural network through interactive attention alignment”. In *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–28.

- Gilpin, Leilani H et al. (2018). “Explaining explanations: An overview of interpretability of machine learning”. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, pp. 80–89.
- Goldstein, Alex et al. (2015). “Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation”. In *Journal of Computational and Graphical Statistics* 24.1, pp. 44–65.
- Hastie, Trevor and Robert Tibshirani (1986). “Generalized additive models”. In *Statistical science* 1.3, pp. 297–310.
- Hendriks, Johannes et al. (2020). “Linearly constrained neural networks”. In *arXiv preprint arXiv:2002.01600*.
- Linardatos, Pantelis, Vasilis Papastefanopoulos, and Sotiris Kotsiantis (2020). “Explainable ai: A review of machine learning interpretability methods”. In *Entropy* 23.1, p. 18.
- Liu, Frederick and Besim Avci (2019). “Incorporating priors with feature attribution on text classification”. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6274–6283.
- Lundberg, Scott M and Su-In Lee (2017). “A unified approach to interpreting model predictions”. In *Advances in neural information processing systems* 30.
- Madsen, Andreas, Siva Reddy, and Sarath Chandar (2022). “Post-hoc interpretability for neural nlp: A survey”. In *ACM Computing Surveys* 55.8, pp. 1–42.
- Molnar, Christoph (2020). “Interpretable machine learning”. Lulu. com.
- Nelder, John Ashworth and Robert WM Wedderburn (1972). “Generalized linear models”. In *Journal of the Royal Statistical Society Series A: Statistics in Society* 135.3, pp. 370–384.
- Pya, Natalya and Simon N Wood (2015). “Shape constrained additive models”. In *Statistics and computing* 25.3, pp. 543–559.
- Qian, Zhaozhi et al. (2021). “Integrating expert ODEs into Neural ODEs: Pharmacology and disease progression”. In *Advances in Neural Information Processing Systems* 34, pp. 11364–11383.
- Rieger, Laura et al. (2020). “Interpretations are useful: penalizing explanations to align neural networks with prior knowledge”. In *International conference on machine learning*. PMLR, pp. 8116–8126.
- Selvaraju, Ramprasaath R et al. (2017). “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.
- Taesiri, Mohammad Reza, Giang Nguyen, and Anh Nguyen (2022). “Visual correspondence-based explanations improve AI robustness and human-AI team accuracy”. In *Advances in Neural Information Processing Systems* 35, pp. 34287–34301.
- Takeishi, Naoya and Alexandros Kalousis (2021). “Physics-integrated variational autoencoders for robust and interpretable generative modeling”. In *Advances in Neural Information Processing Systems* 34, pp. 14809–14821.
- Vellido, Alfredo (2020). “The importance of interpretability and visualization in machine learning for applications in medicine and health care”. In *Neural computing and applications* 32.24, pp. 18069–18083.
- Wehenkel, Antoine and Gilles Louppe (2019). “Unconstrained monotonic neural networks”. In *Advances in neural information processing systems* 32.
- Wu, Mike et al. (2020). “Regional tree regularization for interpretability in deep neural networks”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04, pp. 6413–6421.

- Yeh, I-C (1998). “Modeling of strength of high-performance concrete using artificial neural networks”. In *Cement and Concrete research* 28.12, pp. 1797–1808.
- Yeh, I-Cheng (1998). “Concrete Compressive Strength”. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5PK67>.
- Yin, Yuan et al. (2021). “Augmenting physical models with deep networks for complex dynamics forecasting”. In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.
- Zhang, Quan-shi and Song-Chun Zhu (2018). “Visual interpretability for deep learning: a survey”. In *Frontiers of Information Technology & Electronic Engineering* 19.1, pp. 27–39.
- Zhang, Ye, Iain Marshall, and Byron C Wallace (2016). “Rationale-augmented convolutional neural networks for text classification”. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. Vol. 2016, p. 795.
- Zhang, Yifei et al. (2024). “Megl: Multimodal explanation-guided learning”. In *arXiv preprint arXiv:2411.13053*.
- Zhou, Bolei et al. (2016). “Learning deep features for discriminative localization”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929.

5.A LIST OF VARIABLES

5.A.1 CONCRETE COMPRESSIVE STRENGTH DATASET

Table 5.1. Variables used for the concrete compressive strength dataset. The variable indicated in bold type is the one for which PD prior knowledge is assumed (x_k). The target is indicated in italics.

Name	Description
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)
<i>Strength</i>	<i>Concrete compressive strength [MPa]</i>

5.A.2 PHALK DATASET

Table 5.2. Variables used in the PHALK dataset. The variable indicated in bold type is the one for which PD prior knowledge is assumed (x_k). The target is indicated in italics.

Name	Description
Latitude	Latitude of the basin centroid in WGS84
Longitude	Longitude of the basin centroid in WGS84
Alkalinity	Estimated alkalinity values [μeq/l]
XX_{su}^U	% of unconsolidated sediments within the total watershed upstream of the sub-basin
XX_{ss}^U	% of siliclastic sedimentary rocks within the total watershed upstream of the sub-basin
XX_{sm}^U	% of mixed sedimentary rocks within the total watershed upstream of the sub-basin
XX_{sc}^U	% of carbonate sedimentary rocks within the total watershed upstream of the sub-basin
XX_{py}^U	% of pyroclastics within the total watershed upstream of the sub-basin
XX_{ev}^U	% of evaporates within the total watershed upstream of the sub-basin
XX_{mt}^U	% of metamorphic rocks within the total watershed upstream of the sub-basin
XX_{pa}^U	% of acid plutonic rocks within the total watershed upstream of the sub-basin
XX_{pi}^U	% of intermediate plutonic rocks within the total watershed upstream of the sub-basin
XX_{pb}^U	% of basic plutonic rocks within the total watershed upstream of the sub-basin
XX_{va}^U	% of acid volcanic rocks within the total watershed upstream of the sub-basin
XX_{vi}^U	% of intermediate volcanic rocks within the total watershed upstream of the sub-basin
XX_{vb}^U	% of basic volcanic rocks within the total watershed upstream of the sub-basin
XX_{ig}^U	% of ice and glaciers within the total watershed upstream of the sub-basin
XX_{wb}^U	% of water bodies within the total watershed upstream of the sub-basin
Slope degree	Terrain slope (°) within total watershed upstream of sub-basin
Water surface	1 if basin with water surface
<i>pH</i>	<i>Estimated pH value</i>

6 LEARNING WITH SMALL DATA BY TRANSFERRING MARGINAL EFFECTS FROM LARGE DATA

ADDENDUM

This chapter has previously been published as

Yann Claes, Pierre Geurts, and Vân Anh Huynh-Thu (2026). "Learning with Small Data by Transferring Marginal Effects from Large Data". To appear in proceedings of *International Workshop on Learning from Small Data - ECML-PKDD*.

In the previous chapter, we designed a training approach that considers prior knowledge about model explanations, in the form of the partial dependence function, to enhance both model performance and model alignment with such expert knowledge while being less restrictive than hybrid additive models. Yet, this method requires the definition of a parametric target PD function which may not always be easy to derive depending on the problem structure. This chapter reduces restrictiveness even more by dropping parametric priors and instead assuming access to an additional large database relating a subset of features to the output, alongside a smaller database consisting of complete input-output pairs. Interestingly, the larger database provides information about the marginal plot (M-plot) for the given subset of features, another interpretation technique closely related to the partial dependence function. Therefore, we derive a model and a particular training approach that leverage information about this M-plot to improve model performance when facing scarce high-quality data.

We first introduce the addressed problem in Section 6.1. Then, we formalize the problem statement and methodology in Section 6.2, and provide theoretical elements to motivate our method. In Section 6.3, we cover related lines of work. Section 6.4 evaluates the performance of our approach against related baselines and also conducts an ablation study to estimate the importance of each element used in the method, while Section 6.5 concludes on the limitations of the current study.

Contents

6.1	Introduction	96
6.2	Methodology	97
6.2.1	Problem statement	97
6.2.2	Marginal plots	97
6.2.3	Training approach	98
6.2.4	Asymptotic analysis of the loss function	99
6.3	Related work	100

6.4	Experiments	100
6.4.1	Product problem	101
6.4.2	Real-world problems	101
6.4.3	Ablation study	104
6.5	Conclusion	105

6.1 INTRODUCTION

Over the past decades, practitioners have developed deep learning techniques with increasing applicability to various domains, the critical requirement to all such applications being the availability of abundant training data. Yet, there exist domains for which access to high-quality datasets remains limited, for example in healthcare where large curated datasets are harder to find (Shaikhina et al., 2015; Razzak et al., 2017; Bansal et al., 2022; Upadhyay et al., 2024), being too costly to generate or for privacy reasons.

With scarce training data, deep learning models are more prone to overfitting, and a common solution is to rely on data augmentation. Such techniques are mostly applied to image modalities (Frid-Adar et al., 2018; Sajjad et al., 2019; Lashgari et al., 2020) and they typically enhance both dataset size and quality by leveraging more information from the training data through specific transformations of the input images. However, for problems defined by tabular data, augmentation techniques (Onishi et al., 2023; Cui et al., 2024) remain an unreliable perspective due to the structure and complexity inherent to such data. Indeed, manual enhancing from available table pools does not solve the issue of cost, and generative methods cannot be truly trusted in terms of sample quality.

Another way to improve model robustness and sample efficiency is through the use of prior knowledge, as is done in physics-informed machine learning (ML). For instance, one could assume having access to some first-principles model and impose a particular interaction structure between the prior model and the ML model (Yin et al., 2021; Donà et al., 2022; Claes et al., 2025). Physical properties can be directly integrated into the statistical models, a common example being physics-informed neural networks (PINNs) (Raissi et al., 2019; Farea et al., 2025), or ML models can also be nudged to respect some physical laws by introducing regularization constraints in the loss function (L. Zhang et al., 2018; Jia et al., 2019; Beucler et al., 2021). In the field of explanation-guided learning, prior knowledge can take various forms: feature importance values (Rieger et al., 2020), regions of importance (Mitsuhara et al., 2019; Ismail et al., 2021; Y. Zhang et al., 2024) or even properties that the model should satisfy (Wu et al., 2020; Erion et al., 2021). It has been shown that using explanation knowledge could help improve performance with limited training data (Gao, Sun, Zhao, et al., 2022; Gao, Sun, Bai, et al., 2022). Nevertheless, all these techniques require the definition of prior (algebraic) knowledge about the considered problem, which is not always available.

Instead of parametric prior knowledge, one could have access to auxiliary data characterizing the same problem. This kind of setting is for instance common with two-phase studies (Hidioglou et al., 1998; Chatterjee, Y.-H. Chen, and Breslow, 2003) where inexpensive information on a few covariates is collected for a large population, from which a smaller subset of interest is then selected for the measurement of more detailed covariates. More generally, we assume access to larger amounts of cheap auxiliary data relating a subset of the features to the output, which allows to train a partial model (or alternatively direct access to such a partial model from prior knowledge). In such a setting, we propose a method that fits a model (called the marginal plot, or M-plot, as introduced in (Apley et al., 2020)) on the large incomplete sample, which is then modulated by functions learned from the smaller sample with appropriate regularization such that their combined effect does not deviate too much from the M-plot. On a range of regression problems, both synthetic and real-world, we show that the models obtained with our training approach steadily outperform multiple baselines, and we also evaluate the impact of our modulation and regularization individually.

The rest of the paper is organized as follows. In Section 6.2, we formalize the scope of the problem along with our training approach, which we motivate theoretically. In

Section 6.3, we position our work against related works. In Section 6.4, we empirically demonstrate the impact of our approach and study the effect of both our architecture as well as the regularization constraint.

6.2 METHODOLOGY

In this section, we first formalize the class of problems that we study. Then, we provide some background about marginal plots and derive our training approach.

6.2.1 PROBLEM STATEMENT

Let us define the regression problem,

$$y = f(\mathbf{x}) + \varepsilon,$$

with $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^d$, $d \in \mathbb{N}_+$, drawn from a distribution $p(\mathbf{x}, y)$ and let $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ be some Gaussian noise. Let us assume that we have access to two different databases characterizing the same problem,

- $LS_l = \{(\mathbf{x}_{i,k}, y_i)\}_{i=1}^{N_l}$,
- $LS_s = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_s}$,

with $N_l \gg N_s$ and where $\mathbf{x}_k \subset \mathbf{x}$ is some subset of features. Both LS_l and LS_s are drawn from $p(\mathbf{x}, y)$, i.e., LS_l can be seen as a dataset in which only some covariates $\mathbf{x}_k$ have been measured due to time/cost issues. Let us also assume some differentiable ML model $h^\theta(\mathbf{x})$ where $\theta \in \mathbb{R}^{d_\theta}$ is a trainable parameter vector, e.g., a multilayer perceptron (MLP). We want to learn θ^* such that,

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{\mathbf{x}, y} [(h^\theta(\mathbf{x}) - y)^2]. \quad (6.1)$$

6.2.2 MARGINAL PLOTS

Definition 2. Let $\mathbf{x}_k$ be some subset of features of interest and $\mathbf{x}_{-k}$ its complement, with $\mathbf{x}_k \cup \mathbf{x}_{-k} = \mathbf{x}$, then the marginal plot (M-plot) of a function $f(\mathbf{x})$ on $\mathbf{x}_k$ is defined as (Apley et al., 2020),

$$\begin{aligned} M(f, \mathbf{x}_k) &= \mathbb{E}_{\mathbf{x}_{-k} | \mathbf{x}_k} [f(\mathbf{x}_k, \mathbf{x}_{-k})], \\ &= \int f(\mathbf{x}_k, \mathbf{x}_{-k}) p(\mathbf{x}_{-k} | \mathbf{x}_k) d\mathbf{x}_{-k}, \end{aligned} \quad (6.2)$$

with $p(\mathbf{x}_{-k} | \mathbf{x}_k)$ the conditional distribution of $\mathbf{x}_{-k}$ given $\mathbf{x}_k$.

In other words, the M-plot of a model f given certain values for a subset of features $\mathbf{x}_k$ describes the average response of the model at those values.

Taking back the regression problem from the previous section, one can show that $M(f, \mathbf{x}_k)$, i.e. the M-plot of the true, unknown function f , is the best least-squares model $h_k^*(\mathbf{x}_k)$ regressing y using only $\mathbf{x}_k$ (Hastie, Tibshirani, et al., 2009), defined as:

$$h_k^* = \arg \min_{h_k} \mathbb{E}_{\mathbf{x}, y} [(h_k(\mathbf{x}_k) - y)^2]. \quad (6.3)$$

Indeed, we can decompose the joint expectation in Equation (6.3) by conditioning on $\mathbf{x}_k$,

$$\mathbb{E}_{\mathbf{x}, y} [(h_k(\mathbf{x}_k) - y)^2] = \mathbb{E}_{\mathbf{x}_k} [\mathbb{E}_{y | \mathbf{x}_k} [(h_k(\mathbf{x}_k) - y)^2]]. \quad (6.4)$$

To minimize the global expectation, it is sufficient to minimize the inner conditional expectation in Equation (6.4) for every possible value of $\mathbf{x}_k$. We can write,

$$\begin{aligned} & \mathbb{E}_{y|\mathbf{x}_k} [(h_k(\mathbf{x}_k) - y)^2] \\ &= \mathbb{E}_{y|\mathbf{x}_k} [((h_k(\mathbf{x}_k) - \mathbb{E}_{y|\mathbf{x}_k}[y]) + (\mathbb{E}_{y|\mathbf{x}_k}[y] - y))^2], \\ &= \mathbb{E}_{y|\mathbf{x}_k} [(h_k(\mathbf{x}_k) - \mathbb{E}_{y|\mathbf{x}_k}[y])^2] + \mathbb{E}_{y|\mathbf{x}_k} [(\mathbb{E}_{y|\mathbf{x}_k}[y] - y)^2] \\ &\quad + 2 \mathbb{E}_{y|\mathbf{x}_k} [(h_k(\mathbf{x}_k) - \mathbb{E}_{y|\mathbf{x}_k}[y])(\mathbb{E}_{y|\mathbf{x}_k}[y] - y)]. \end{aligned}$$

We can notice that the cross-product vanishes (the first factor is a constant given $\mathbf{x}_k$ and the second is zero), whereas the first term is a constant given $\mathbf{x}_k$. Hence, the equation simplifies to,

$$\mathbb{E}_{y|\mathbf{x}_k} [(h_k(\mathbf{x}_k) - y)^2] = (h_k(\mathbf{x}_k) - \mathbb{E}_{y|\mathbf{x}_k}[y])^2 + \mathbb{E}_{y|\mathbf{x}_k} [(\mathbb{E}_{y|\mathbf{x}_k}[y] - y)^2].$$

The second term corresponds to the variance of y given $\mathbf{x}_k$ and is thus irreducible. To minimize the mean-squared error, we must choose an $h_k(\mathbf{x}_k)$ that forces the first term towards zero, i.e.,

$$h_k^*(\mathbf{x}_k) = \mathbb{E}_{y|\mathbf{x}_k}[y].$$

Given that $y = f(\mathbf{x}) + \epsilon$ and $\mathbb{E}_{\epsilon|\mathbf{x}_k}[\epsilon] = 0$, we can write,

$$\begin{aligned} h_k^*(\mathbf{x}_k) &= \mathbb{E}_{\mathbf{x}_{-k}|\mathbf{x}_k} [f(\mathbf{x}_k, \mathbf{x}_{-k})], \\ &= M(f, \mathbf{x}_k). \end{aligned} \tag{6.5}$$

6.2.3 TRAINING APPROACH

We start by fitting a model $h_k^\phi(\mathbf{x}_k)$ on LS_l , with $\phi \in \mathbb{R}^{d_\phi}$, hence obtaining

$$\hat{h}_k^\phi(\mathbf{x}_k) \approx M(f, \mathbf{x}_k),$$

given that LS_l is of finite size. We then leverage this large-sample M-plot estimate in two ways. First, we fit on LS_s a model $h^\theta(\mathbf{x})$ of the following form, inspired by (Perez et al., 2018):

$$h^\theta(\mathbf{x}) = (1 + \gamma^{\theta_\gamma}(\mathbf{x})) \cdot \hat{h}_k^\phi(\mathbf{x}_k) + \beta^{\theta_\beta}(\mathbf{x}). \tag{6.6}$$

where γ and β respectively act as *scaling* and *shifting* functions. The choice of $(1 + \gamma^{\theta_\gamma})$ ensures that the model is initialized close to the M-plot estimate when γ and β are initialized close to 0, which constitutes a decent starting point when only few data samples are available.

Second, as a way to reduce overfitting, we would like to constrain the shape of the M-plot of the final model h^θ to be close to $\hat{h}_k^\phi$, i.e., we would like,

$$\begin{aligned} \forall \mathbf{x}_k : \mathbb{E}_{\mathbf{x}_{-k}|\mathbf{x}_k} [h^\theta(\mathbf{x}_k, \mathbf{x}_{-k})] &= \hat{h}_k^\phi(\mathbf{x}_k), \\ &\approx M(f, \mathbf{x}_k), \end{aligned} \tag{6.7}$$

which is equivalent to minimizing, for all $\mathbf{x}_k$,

$$(\mathbb{E}_{\mathbf{x}_{-k}|\mathbf{x}_k} [h^\theta(\mathbf{x}_k, \mathbf{x}_{-k})] - \hat{h}_k^\phi(\mathbf{x}_k))^2.$$

However, in a finite-sample setting, estimating the left-hand side of Equation (6.7) is generally impossible as we do not have access to enough pairs to reasonably approximate the conditional expectation. We therefore propose to regularize the shape of h^θ by penalizing *sample-wise* deviations from the M-plot estimator $\hat{h}_k^\phi$, i.e.,

$$\mathcal{L}_M(h^\theta, \hat{h}_k^\phi) = \mathbb{E}_{\mathbf{x}} [(h^\theta(\mathbf{x}_k, \mathbf{x}_{-k}) - \hat{h}_k^\phi(\mathbf{x}_k))^2], \tag{6.8}$$

which gives the final training objective,

$$\mathcal{L}(h^\theta, \hat{h}_k^\phi, y) = \mathcal{L}_{mse}(h^\theta, y) + \lambda_M \mathcal{L}_M(h^\theta, \hat{h}_k^\phi), \quad (6.9)$$

where $\mathcal{L}_{mse} = \mathbb{E}_{\mathbf{x}, y}[(h^\theta(\mathbf{x}) - y)^2]$ and λ_M is a regularization coefficient. In practice, both $\mathcal{L}_{mse}$ and $\mathcal{L}_M$ are estimated through sample means computed on the small dataset LS_s , respectively,

$$\hat{\mathcal{L}}_{mse}(h^\theta, y) = \frac{1}{N_s} \sum_{i=1}^{N_s} (h^\theta(\mathbf{x}_i) - y_i)^2, \quad (6.10)$$

$$\hat{\mathcal{L}}_M(h^\theta, \hat{h}_k^\phi) = \frac{1}{N_s} \sum_{i=1}^{N_s} (h^\theta(\mathbf{x}_i) - \hat{h}_k^\phi(\mathbf{x}_{i,k}))^2. \quad (6.11)$$

6.2.4 ASYMPTOTIC ANALYSIS OF THE LOSS FUNCTION

The following two results characterize the solution of Equation (6.9) in the asymptotic (population) setting, i.e., assuming infinite training sample sizes ($N_s = N_l = \infty$).

Theorem 1. *The minimizer of $\mathcal{L}$ is $h^*(\mathbf{x}) = \frac{f(\mathbf{x}) + \lambda_M M(f, \mathbf{x}_k)}{1 + \lambda_M}$.*

Proof. We are looking for $h(\mathbf{x})$ that minimizes,

$$\mathbb{E}_{\mathbf{x}} \left[\mathbb{E}_{y|\mathbf{x}}[(h(\mathbf{x}) - y)^2] + \lambda_M (h(\mathbf{x}) - \hat{h}_k^\phi(\mathbf{x}_k))^2 \right].$$

In the asymptotic setting, note that $\hat{h}_k^\phi(\mathbf{x}_k) = M(f, \mathbf{x}_k)$. Assuming that we can choose each value $h(\mathbf{x})$ independently, we want to find, for all $\mathbf{x}$,

$$h^*(\mathbf{x}) = \arg \min_{h(\mathbf{x})} \left[\mathbb{E}_{y|\mathbf{x}}[(h(\mathbf{x}) - y)^2] + \lambda_M (h(\mathbf{x}) - M(f, \mathbf{x}_k))^2 \right].$$

If we cancel the derivative of the loss according to $h(\mathbf{x})$, one gets,

$$\mathbb{E}_{y|\mathbf{x}}[2(h^*(\mathbf{x}) - y)] + \lambda_M 2(h^*(\mathbf{x}) - M(f, \mathbf{x}_k)) = 0,$$

which yields,

$$h^*(\mathbf{x}) = \frac{f(\mathbf{x}) + \lambda_M M(f, \mathbf{x}_k)}{1 + \lambda_M}. \quad (6.12)$$

□

As could be expected from a regularization constraint, we can notice that the optimal model $h^*(\mathbf{x})$ will be pushed towards $M(f, \mathbf{x}_k)$ for larger values of λ_M .

Corollary 1. *The M-plot of the minimizer $h^*(\mathbf{x})$ is exactly $M(f, \mathbf{x}_k)$.*

Proof. Recall the definition of $h^*(\mathbf{x})$,

$$h^*(\mathbf{x}) = \frac{f(\mathbf{x}) + \lambda_M M(f, \mathbf{x}_k)}{1 + \lambda_M}.$$

The M-plot of $h^*(\mathbf{x})$ is given by,

$$\begin{aligned} M(h^*, \mathbf{x}_k) &= \frac{1}{1 + \lambda_M} (M(f, \mathbf{x}_k) + \lambda_M M(M(f, \mathbf{x}_k))), \\ &= \frac{1}{1 + \lambda_M} (M(f, \mathbf{x}_k) + \lambda_M M(f, \mathbf{x}_k)), \\ &= M(f, \mathbf{x}_k). \end{aligned} \quad (6.13)$$

□

We thus observe that, asymptotically, training a model to minimize Equation (6.9) actually fulfills the desired constraint defined in Equation (6.7), whatever the value of λ_M .

6.3 RELATED WORK

Our framework is conceptually close to varying-coefficient models (Hastie and Tibshirani, 1993), if we were to remove $\mathbf{x}_k$ from the modulating functions. Indeed, the resulting model $h(\mathbf{x}) = (1 + \gamma(\mathbf{x}_{-k})) \cdot h_k(\mathbf{x}_k) + \beta(\mathbf{x}_{-k})$ would then be an affine transformation of $h_k(\mathbf{x}_k)$. The main difference is that for varying-coefficient models, $h_k(\mathbf{x}_k)$ would rather be a fixed set of predictors, whereas it is a learned model obtained from a large database in our approach. Furthermore, our method comes with regularization constraints to reduce overfitting.

Several works studied problems with two datasets of different sizes and features. In the setting of two-phase studies, Chatterjee, Y.-H. Chen, Maas, et al. (2016) tackle maximum likelihood estimation (MLE) of a model $f(y \mid x, z)$ from a small internal study with complete covariates (x, z, y) , by exploiting the parameters of a big-data model $g_\theta(y \mid x)$. However, their constraint controls f through a score function defined from g_θ , whereas ours enforces M-plot preservation through least-squares regularization of the modulating functions. A. Chen et al. (2015) consider a two-database setting but the larger database is composed of biased outputs rather than partial observations. Similarly, Jin et al. (2023) leverage conditional independence structure by using an auxiliary dataset, yet both works tackle a very different problem setting and are limited to linear models.

If we consider h_k to be a physical signal, then enforcing the structure of the model to be h_k modulated by γ and β amounts to traditional residual learning methods (setting γ to 0) as done in (Yin et al., 2021; Donà et al., 2022; Claes et al., 2025). If h_k represented some physical property about the inputs $\mathbf{x}_k$, then regularization towards h_k could be seen as enforcing the model to respect such properties (L. Zhang et al., 2018; Jia et al., 2019; Beuclet et al., 2021). However, such methods rely on the definition of a parametric prior physical model. More generally, our approach can be linked to hybrid physics-data models (Parish et al., 2016; Willard et al., 2020; Daw et al., 2022), if h_k was to be combined with the modulators differently than through the identity mapping, for instance by being fed as input to the data-driven model, as introduced in stacked generalization (Wolpert, 1992). We will include residual learning and stacking as baselines in our experiments in the next section.

6.4 EXPERIMENTS

In this section, we compare on several regression problems (both synthetic and real-world) the predictive performance of our modulating approach (named hereafter "Modulated (C)") to the following baselines:

- Stacked learning, i.e. an MLP taking as additional input the output of $\hat{h}_k^\phi$ and trained with the standard MSE loss $\mathcal{L}_{mse}$,
- $\hat{h}_k^\phi$, the M-plot estimator,
- MLP (C), i.e. when h^θ is a standard MLP trained with the regularized loss $\mathcal{L} = \mathcal{L}_{mse} + \lambda_M \mathcal{L}_M$,
- An MLP trained with the loss $\mathcal{L}_{mse}$, but regularized with dropout ($p = 0.3$) and weight decay ($\lambda = 0.01$).

All MLPs except γ^{θ_γ} and β^{θ_β} are 3-layered MLPs of decreasing hidden layer sizes (128, 64, and 32 units) whereas γ^{θ_γ} and β^{θ_β} are set to MLPs of 2 layers (16 and 8 units). MLPs are implemented using PyTorch (Paszke et al., 2019). For all models, we optimize the learning rate in $\{5 \times 10^{-2}, 5 \times 10^{-3}\}$, and for all regularized approaches, we optimize λ_M in $\{0.15, 0.25, 0.5, 0.75, 1, 1.5\}$.

For each real-world problem, we fit a random forest model (Breiman, 2001) using scikit-learn (Pedregosa et al., 2011) to evaluate feature importances. We then select one or a few important variables to constitute $\mathbf{x}_k$, depending on the total number of available features, create datasets accordingly, and standardize both inputs and outputs. In order to select the optimal model, for each baseline, we evaluate $\hat{\mathcal{L}}_{mse}$ on a validation set of size equal to the corresponding training set. Eventually, we evaluate the predictive performance of all baselines by estimating $\hat{\mathcal{L}}_{mse}$ on a test set.

6.4.1 PRODUCT PROBLEM

We consider the following problem,

$$y = 15x_0^3 \cos(\pi x_0) \frac{(x_1 - 1)^2}{2} + \sum_{j=2}^9 0x_j + \varepsilon,$$

where features are correlated with zero mean, variances and covariances equal to 0.5 and 0.25, respectively, and $\varepsilon \sim \mathcal{N}(0, 1)$. We set the size of the large dataset to $N_l = 1500$ samples, and construct the latter as $LS_l = \{(x_0, y)\}_{i=1}^{N_l}$. For the small database, we study different training sizes N_s , namely 150, 100 and 50. We fix the size of the test set to 10 000 samples.

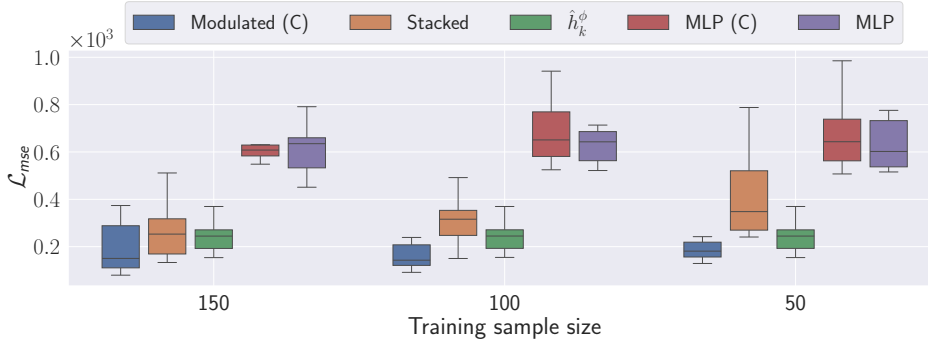


Figure 6.1. Evolution of $\mathcal{L}_{mse}$ w.r.t. training sample size, on the product problem. Boxplots summarize the results on the test set, over 10 randomly selected datasets.

We observe in Figure 6.1 that all the baselines that leverage information from LS_l perform better than the uninformed MLP (purple), except for the MLP constrained w.r.t. $\hat{h}_k^\phi$ (red). This observation suggests that using $\mathcal{L}_M$ as sole regularization technique to train an MLP from scratch is not sufficient to learn a model robust to data scarcity conditions. The M-plot estimator $\hat{h}_k^\phi$ is actually a well-performing model, which is due to the fact that the most important part of the input signal depends on x_0 . Finally, we observe that our modulated architecture outperforms all other approaches, indicating that it better grasps the interaction between x_0 and x_1 .

6.4.2 REAL-WORLD PROBLEMS

For all problems, we set the size of the large dataset to $N_l = 5000$ randomly selected samples. For the small database, we compare the performance when training on $N_s = 150, 100$ and 50 training samples (also randomly selected). The remaining samples serve as test set. We first provide a brief description of each problem and the features selected as $\mathbf{x}_k$, then we discuss the obtained results.

DATASETS

Beijing housing dataset This dataset contains features related to transactions in the housing market, whose goal is to predict the total price of a house (lianjia.com, 2017). In particular, it includes information related to the architecture (number of living rooms, kitchens, etc.) as well as some supplementary information about the transaction. We decided to include all features concerning the parts of the house (number of rooms of each type), the square footage, along with its latitude and longitude, the number of active days on the market and the number of followers of the transaction, for a total of 10 features. For this problem, we used the two most important features as $\mathbf{x}_k$, i.e., the square footage of the house along with its latitude.

Bike sharing demand dataset This dataset collects weather and seasonal features with hourly counts of bike rentals (Fanaee-T, 2013). We use the dataset version curated for regression as presented in (Grinsztajn et al., 2022), containing six numerical features: hour and month as time-related features, and wind speed, relative humidity, temperature and feels-like temperature as weather features. In this scenario, we used the hour as covariate $\mathbf{x}_k$, the latter also being the variable of highest importance.

Miami housing dataset This dataset is composed of samples of single-family homes sold in Miami in 2016 (Mayer et al., 2022), relating living area to their sale price, along with additional location and distance information, for a total of 14 features (Grinsztajn et al., 2022). We used the three most important features as $\mathbf{x}_k$, namely the distance to the nearest sub-center, the distance to the business district and the total floor area.

Superconduct dataset This dataset contains quantitative values about properties of superconductors (Hamidieh, 2018), for a total of 79 features, and the goal is to predict the critical temperature of a given superconductor (Grinsztajn et al., 2022). We decided to use the three most important variables as $\mathbf{x}_k$.

Ailerons dataset In this problem, the goal is to predict the control action on the ailerons of an F16 aircraft (Torgo et al., 2014), from a series of attributes (33) describing the state of the plane (Grinsztajn et al., 2022). As done before, we decided to use the three most important variables as $\mathbf{x}_k$.

DISCUSSION OF THE RESULTS

We can observe in Figure 6.2 that with 150 training samples, our regularized modulated approach (blue) performs systematically better than a classical MLP with standard regularization (purple), except on the ailerons dataset where both seem to perform equivalently. The latter case indicates that there seems to be enough training points to train a decent model with no further regularization. Furthermore, our method generally outperforms stacked modeling (orange), or at least performs on par. Still, the greater gap between the performance of both methods on the superconduct and ailerons datasets suggests that the higher the number of features, the less robust are the models obtained with stacked learning. It is also worth noting that the M-plot estimator $\hat{h}_k^\phi$ (green) consistently performs worse than our approach. For all training sizes, the MLP regularized with $\mathcal{L}_M$ (red) mostly performs worse than the latter, which highlights that $\mathcal{L}_M$ is not powerful enough alone.

With 100 training samples, the gap w.r.t. the uninformed MLP (purple) now looks clearer, and the difference in performance against stacked modeling remains similar. Interestingly, the performance of our model on the ailerons dataset is roughly identical to its performance after training on 150 training samples, which is definitely not the

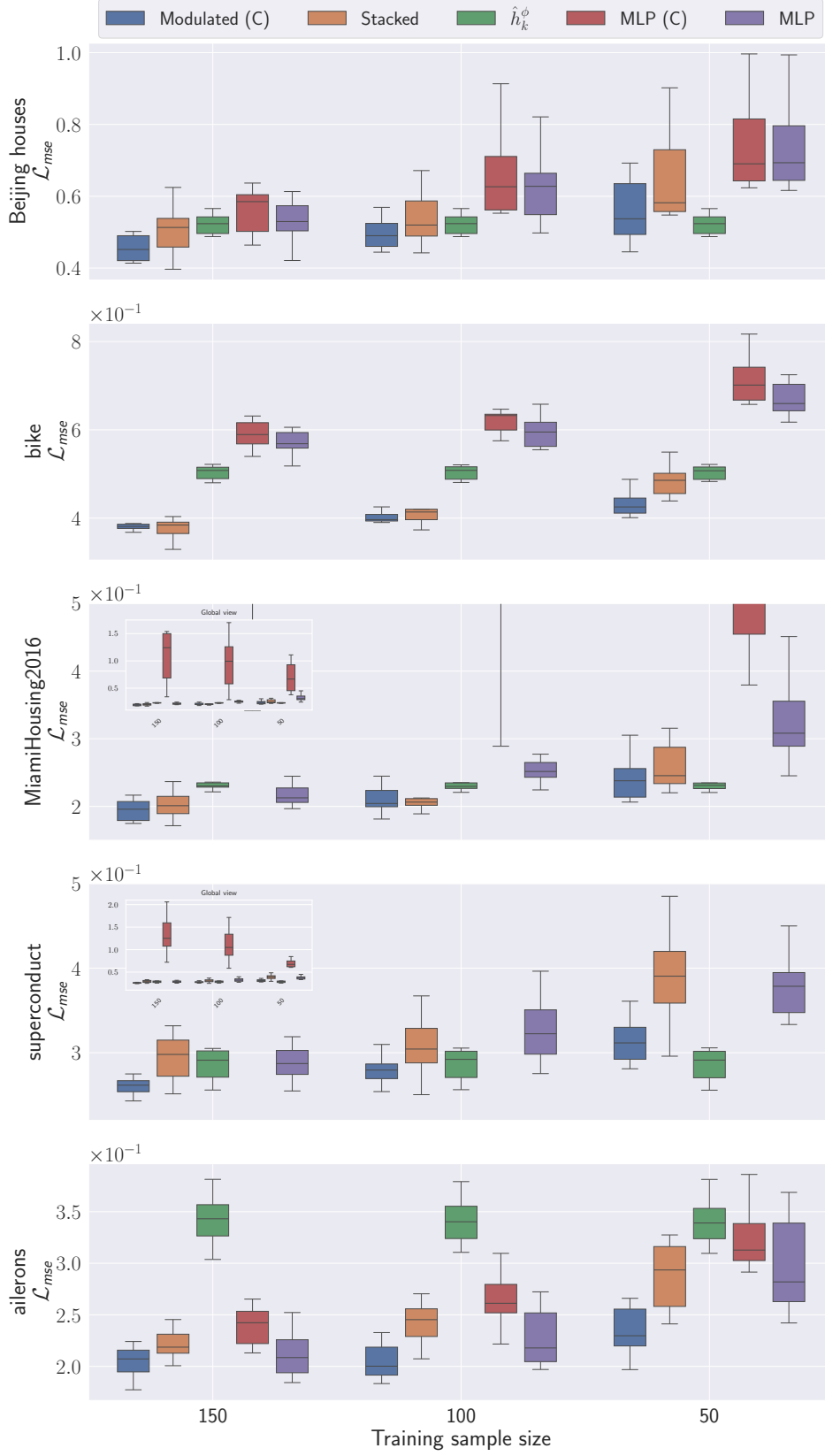


Figure 6.2. Evolution of $\mathcal{L}_{mse}$ w.r.t. training sample size, on all real-world problems. Boxplots summarize results on the test set, over 10 randomly selected datasets.

case for stacked learning nor for the other baselines that all suffered a decrease in performance.

Finally, under very scarce data conditions (50 training samples), we can observe that our method generally outperforms all other baselines (the performance gap with stacked modeling being also easier to notice). An exception is the M-plot estimator which performs better or on par with our regularized modulated approach on some problems, such as the superconduct problem. For the latter, this could be explained by the fact that in this specific setting, the number of features is greater than the number of training samples. That being said, while with 50 training points $\hat{h}_k^\phi$ could sometimes be a better choice, this is certainly not the case with larger training set sizes.

6.4.3 ABLATION STUDY

In this section, we study the impact of both our modulated architecture as well as the introduction of the regularization constraint $\mathcal{L}_M$ in the training loss. For that purpose, we consider the following baselines:

- Residual learning, where $h^\theta(\mathbf{x}) = \hat{h}_k^\phi(\mathbf{x}_k) + \beta^{\theta_\beta}(\mathbf{x})$, trained with $\mathcal{L}_{mse}$,
- Residual learning (C), trained with $\mathcal{L} = \mathcal{L}_{mse} + \lambda_M \mathcal{L}_M$,
- Modulated learning, where $h^\theta(\mathbf{x}) = (1 + \gamma^{\theta_\gamma}(\mathbf{x})) \cdot \hat{h}_k^\phi(\mathbf{x}_k) + \beta^{\theta_\beta}(\mathbf{x})$, trained with $\mathcal{L}_{mse}$,
- $\hat{h}_k^\phi$, the M-plot estimator.

To limit the amount of discussion, we decided to train models on 100 training samples, as it represents an intermediate training size encountered in the previous experiments.

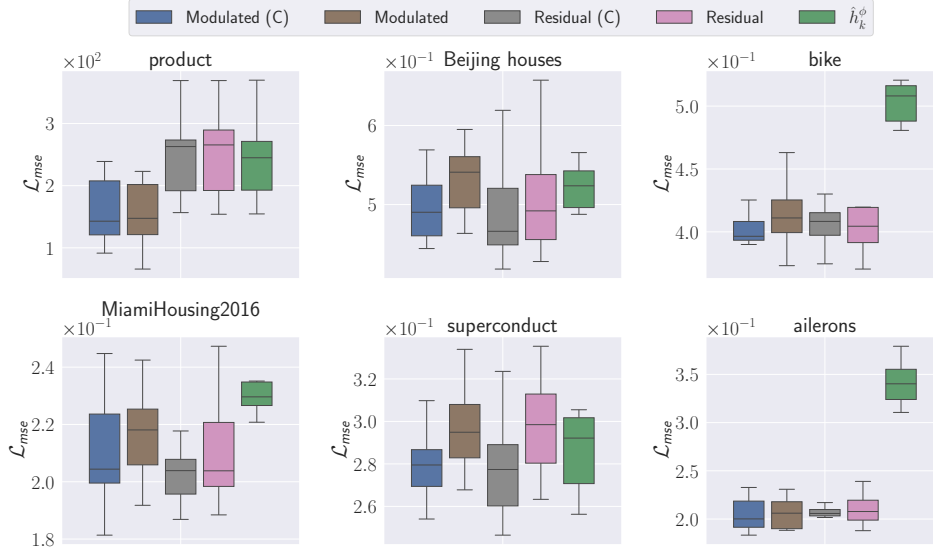


Figure 6.3. Evolution of $\mathcal{L}_{mse}$ on all problems. Boxplots summarize results on the test set, over 10 randomly selected datasets, for a training size of 100 samples.

In Figure 6.3, we can observe that constraining the model with $\mathcal{L}_M$ (blue and grey) helps to reduce errors in comparison with the non-regularized counterparts (brown and pink). For both the product and ailerons problems, the constraint yields equally-performing models, which could indicate that models have access to sufficient training data not to require regularization. The modulation architecture seems necessary for the product problem,

as we observe that residual learning does not perform better than the M-plot estimator. Regarding the other problems, modulation does not seem as important. Still, for the superconduct dataset, avoiding the multiplicative modulation appears to perform slightly better, whereas for the bike problem, the modulation seems slightly more appropriate. Results on the Miami housing dataset tend to indicate that the problem structure could be additive.

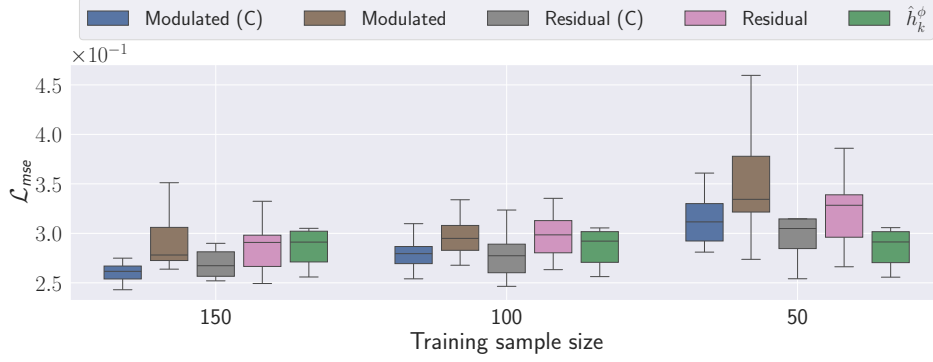


Figure 6.4. Evolution of $\mathcal{L}_{mse}$ w.r.t. training sample size, on the superconduct problem. Boxplots summarize results on the test set, over 10 randomly selected datasets.

In Figure 6.4, we display the evolution of the modulated and regularized baselines on the superconduct problem for each training sample size. With even scarcer training data (50), we notice that the regularized residual model still mildly outperforms our regularized modulated approach; but, more importantly, we observe that neither modulated nor residual learning perform better than the M-plot estimator. However, with a few more samples (150), the trend is inverted as our approach is now the best-performing method. This can be explained by looking at the performance of the M-plot estimator $\hat{h}_k^\phi$. Indeed, it is already a good approximator of the true data relationship as could also be observed in Figure 6.2. Hence, with too few training points (and even more with that many features) the modulating functions rather fit noise than the true signal, hence residual learning methods may perform better as they are actually more regularized by construction. Nevertheless, adding samples to the training data seems to favor the modulated architecture.

6.5 CONCLUSION

In a setting where one has access to large amounts of auxiliary data on a restricted subset of inputs but limited complete training data, we derive a method that fits a model on the former (the marginal plot) and modulates its output on the latter through specific shifting and scaling functions regularized to preserve the marginal plot. On a series of regression problems, we empirically validate the benefits of such approach under varying conditions of data scarcity. We also study the interest of both modulation and M-plot regularization. On the one hand, we conclude that for architectures using the M-plot as a backbone, regularization is a central part to reduce overfitting. On the other hand, the type of modulation, i.e., strictly additive or multiplicative, depends on the unknown structure of the data, although the latter has proven to be of interest in some problems.

While we have only tested our approach by fitting a marginal model on the large training sample, the framework remains entirely valid if one has instead access to prior knowledge in the form of a predictive model based on a subset of features of interest. The performance of the final model would then be dependent on the quality of this prior model, and it would

be interesting to validate this idea on practical problems. As future work, we would be interested in conducting a thorough study and comparison of the different combinations of modulation and regularization against training set size, number of features of interest and their nature (importance) in order to fully characterize the method along with its working conditions. Furthermore, it would have been insightful to compare to classical transfer learning techniques that would fine-tune the M-plot estimator, but it would also be interesting to study transfer learning approaches combined with this modulation setting.

REFERENCES

- Apley, Daniel W and Jingyu Zhu (2020). “Visualizing the effects of predictor variables in black box supervised learning models”. In *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.4, pp. 1059–1086.
- Bansal, Ms Aayushi, Dr Rewa Sharma, and Dr Mamta Kathuria (2022). “A systematic review on data scarcity problem in deep learning: solution and applications”. In *ACM Computing Surveys (Csur)* 54.10s, pp. 1–29.
- Beucler, Tom et al. (2021). “Enforcing analytic constraints in neural networks emulating physical systems”. In *Physical review letters* 126.9, p. 098302.
- Breiman, Leo (2001). “Random forests”. In *Machine learning* 45.1, pp. 5–32.
- Chatterjee, Nilanjan, Yi-Hau Chen, and Norman E Breslow (2003). “A pseudoscore estimator for regression problems with two-phase sampling”. In *Journal of the American Statistical Association* 98.461, pp. 158–168.
- Chatterjee, Nilanjan, Yi-Hau Chen, Paige Maas, et al. (2016). “Constrained maximum likelihood estimation for model calibration using summary-level information from external big data sources”. In *Journal of the American Statistical Association* 111.513, pp. 107–117.
- Chen, Aiyou, Art B. Owen, and Minghui Shi (2015). “Data enriched linear regression”. In *Electronic Journal of Statistics* 9.1, pp. 1078–1112.
- Claes, Yann, Vân Anh Huynh-Thu, and Pierre Geurts (2025). “Hybrid additive modeling with partial dependence for supervised regression and dynamical systems forecasting”. In *Machine Learning* 114.3, p. 72.
- Cui, Lingxi et al. (2024). “Tabular data augmentation for machine learning: Progress and prospects of embracing generative ai”. In *ACM Computing Surveys*.
- Daw, Arka et al. (2022). “Physics-guided neural networks (pgnn): An application in lake temperature modeling”. In *Knowledge guided machine learning*. Chapman and Hall/CRC, pp. 353–372.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Erion, Gabriel et al. (2021). “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”. In *Nature machine intelligence* 3.7, pp. 620–631.
- Fanaee-T, Hadi (2013). “Bike Sharing”. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5W894>.
- Farea, Amer, Olli Yli-Harja, and Frank Emmert-Streib (2025). “Using Physics-Informed Neural Networks for Modeling Biological and Epidemiological Dynamical Systems”. In *Mathematics* 13.10, p. 1664.
- Frid-Adar, Maayan et al. (2018). “GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification”. In *Neurocomputing* 321, pp. 321–331.
- Gao, Yuyang, Tong Steven Sun, Guangji Bai, et al. (2022). “Res: A robust framework for guiding visual explanation”. In *proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 432–442.

- Gao, Yuyang, Tong Steven Sun, Liang Zhao, et al. (2022). “Aligning eyes between humans and deep neural network through interactive attention alignment”. In *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2, pp. 1–28.
- Grinsztajn, Léo, Edouard Oyallon, and Gaël Varoquaux (2022). “Why do tree-based models still outperform deep learning on typical tabular data?”. In *Advances in Neural Information Processing Systems* 35, pp. 507–520.
- Hamidieh, Kam (2018). “Superconductivity Data”. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C53P47>.
- Hastie, Trevor and Robert Tibshirani (1993). “Varying-coefficient models”. In *Journal of the Royal Statistical Society Series B: Statistical Methodology* 55.4, pp. 757–779.
- Hastie, Trevor, Robert Tibshirani, et al. (2009). “The elements of statistical learning: data mining, inference, and prediction”. Vol. 2. Springer.
- Hidiroglou, Mike A and Carl-Erik Särndal (1998). “Use of auxiliary information for two-phase sampling”. In *Survey Methodology* 24, pp. 11–20.
- Ismail, Aya Abdelsalam, Hector Corrada Bravo, and Soheil Feizi (2021). “Improving deep learning interpretability by saliency guided training”. In *Advances in Neural Information Processing Systems* 34, pp. 26726–26739.
- Jia, Xiaowei et al. (2019). “Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles”. In *Proceedings of the 2019 SIAM international conference on data mining*. SIAM, pp. 558–566.
- Jin, Ying and Dominik Rothenhäusler (2023). “Modular regression: improving linear models by incorporating auxiliary data”. In *Journal of Machine Learning Research* 24.351, pp. 1–52.
- Lashgari, Elnaz, Dehua Liang, and Uri Maoz (2020). “Data augmentation for deep-learning-based electroencephalography”. In *Journal of Neuroscience Methods* 346, p. 108885.
- lianjia.com (2017). “Housing price in Beijing”. Kaggle.
- Mayer, Michael and Steven C. Bourassa (2022). “MiamiHousing2016”. OpenML Dataset ID: 43093.
- Mitsuhara, Masahiro et al. (2019). “Embedding human knowledge into deep neural network via attention map”. In *arXiv preprint arXiv:1905.03540*.
- Onishi, Soma and Shoya Meguro (2023). “Rethinking data augmentation for tabular data in deep learning”. In *arXiv preprint arXiv:2305.10308*.
- Parish, Eric J and Karthik Duraisamy (2016). “A paradigm for data-driven predictive modeling using field inversion and machine learning”. In *Journal of computational physics* 305, pp. 758–774.
- Paszke, Adam et al. (2019). “Pytorch: An imperative style, high-performance deep learning library”. In *Advances in neural information processing systems* 32.
- Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Perez, Ethan et al. (2018). “Film: Visual reasoning with a general conditioning layer”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32 (1).
- Raissi, Maziar, Paris Perdikaris, and George E Karniadakis (2019). “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In *Journal of Computational physics* 378, pp. 686–707.

- Razzak, Muhammad Imran, Saeeda Naz, and Ahmad Zaib (2017). “Deep learning for medical image processing: Overview, challenges and the future”. In *Classification in BioApps: Automation of decision making*, pp. 323–350.
- Rieger, Laura et al. (2020). “Interpretations are useful: penalizing explanations to align neural networks with prior knowledge”. In *International conference on machine learning*. PMLR, pp. 8116–8126.
- Sajjad, Muhammad et al. (2019). “Multi-grade brain tumor classification using deep CNN with extensive data augmentation”. In *Journal of computational science* 30, pp. 174–182.
- Shaikhina, Torgyn et al. (2015). “Machine learning for predictive modelling based on small data in biomedical engineering”. In *IFAC-PapersOnLine* 48.20, pp. 469–474.
- Torgo, Luís and Rui Camacho (2014). “Ailerons”. OpenML Dataset ID: 296.
- Upadhyay, Ashwini Kumar and Ashish Kumar Bhandari (2024). “Advances in deep learning models for resolving medical image segmentation data scarcity problem: a topical review”. In *Archives of Computational Methods in Engineering* 31.3, pp. 1701–1719.
- Willard, Jared et al. (2020). “Integrating physics-based modeling with machine learning: A survey”. In *arXiv preprint arXiv:2003.04919* 1.1, pp. 1–34.
- Wolpert, David H (1992). “Stacked generalization”. In *Neural networks* 5.2, pp. 241–259.
- Wu, Mike et al. (2020). “Regional tree regularization for interpretability in deep neural networks”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04, pp. 6413–6421.
- Yin, Yuan et al. (2021). “Augmenting physical models with deep networks for complex dynamics forecasting”. In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.
- Zhang, Linfeng et al. (2018). “Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics”. In *Physical review letters* 120.14, p. 143001.
- Zhang, Yifei et al. (2024). “Megl: Multimodal explanation-guided learning”. In *arXiv preprint arXiv:2411.13053*.

III

EPILOGUE

7 DISCUSSION

Our imagination is stretched to the utmost, not, as in fiction, to imagine things which are not really there, but just to comprehend those things which are there.

— Richard Feynman (1965)

In the introductory paragraphs of Chapter 1, we have tried to emphasize the importance of *models*, or more generally of *modeling assumptions*, in the context of modern problem-solving tasks. While models most often come with approximations or stringent assumptions, they have been studied and improved upon for centuries. When provided with mathematical or physical models describing a given task, a crucial question is to decide on how to use this abstraction. Should it be considered as a perfect model of truth, i.e., a rigid representation of some process inheriting from years of knowledge? Should it instead be sent off the table due to potentially misplaced inherent biases or due to efficiency-related issues? Hence, should we fully rely on data-driven (machine learning) models?

The fundamental idea of this thesis was to study how first-principles models (in various forms) could be combined with machine learning models, and how the nature of this interaction influenced both the performance of the final model along with its physical plausibility. For that purpose, several training methods have been developed, each of them making different assumptions on the modeling framework and on the source of prior knowledge. In this chapter, we discuss the different findings of this work and reflect on future work and perspectives. In particular, Sections 7.1 and 7.2 discuss our contributions to the field of hybrid additive modeling, as presented in Chapters 3 and 4. Section 7.3 reviews our attempt to relax the additivity bias through explanation-guided learning, as was thoroughly analyzed in Chapter 5, whereas Section 7.4 covers our approach to leverage knowledge from data, as explored in Chapter 6. Finally, Section 7.5 provides more general perspectives about the broad field of knowledge-guided machine learning while Section 7.6 concludes with the global message carried by the thesis.

7.1 CONFINING HYBRID ADDITIVE MODELS TO STATIC SUPERVISED REGRESSION PROBLEMS

A first striking observation to be made about hybrid additive models, i.e., hybrid models which sum the contributions of a physical and a statistical model, is that they have most often been studied in the context of dynamical systems forecasting problems. Historically, while some early works cover steady-state problems (Kramer et al., 1992; Thompson et al., 1994), they remain very specific to particular process modeling tasks, and other variants focus exclusively on time-series prediction tasks (Su et al., 1993; Duarte et al., 2004; D. S. Lee et al., 2005). Still, none of these works include theoretical aspects relating to the convergence (and quality) of such hybrid additive models. More recently, several methods to regularize these hybrid models have been developed (Yin, Le Guen, et al.,

2021; Donà et al., 2022) with some level of effort to provide theoretical results on the training approach. Yet, the focus remains on dynamical systems forecasting whereas proofs often consider over-simplistic settings, e.g., dynamics with linear first-principles knowledge, and, above all, provide guarantees only on the convergence of the algorithm, not on the quality of its solution.

Therefore, as opposed to this popular *dynamical* systems forecasting setting, we have decided to move the focus and study the behavior of hybrid additive models in a *static* supervised regression setting. In Chapter 3, we provide clear assumptions which define a typical problem setting for which we derive proofs of optimal fitting in the sense of physical estimation faithfulness. In particular, we model the assumptions of additivity (A1), disjoint feature sets (A2) and statistically independent features (A3) to show that both sequential fitting and alternate training require that all three hypotheses are satisfied to recover a physically-sound estimation of the first-principles parameters.

To empirically validate our proposition, we evaluate the performance of these hybrid additive models on different static regression problems. Specifically, we study simple problems that explicitly satisfy or violate the aforementioned assumptions, which helps characterizing the behavior of hybrid models. We also introduce several modeling approaches, e.g., filtering, to further verify our claims. When all three assumptions are met, as is the case with the classical *Friedman* problem (Friedman et al., 1983), we highlight in Table 3.1 that all hybrid methods yield equivalently-performing models, both in terms of final prediction and in the estimation of the first-principles model, as is expected from the theory. However, when dropping A3, i.e., when input features are no longer statistically independent, we show with a simple *correlated linear* problem that both sequential and alternate training fail to recover a physically-sound parameter estimation, as can be observed in Table 3.2, even though their prediction performance is solid. We underline that, for alternate training, it is nonetheless possible to obtain more plausible physical parameter values provided that the machine learning model receives filtered inputs. In fact, in such a setting, although the initial physical approximation is equal to that of sequential fitting, it is nevertheless possible to recover from this misfit throughout training, thanks to the iterative alternate steps. We also confirm this behavior with the *correlated Friedman* problem.

Eventually, we evaluate the impact of violating A2, which can happen when the prior knowledge is ill-defined. In this case, there exists an overlap between the features of the first-principles model and those of the unknown residual terms. Through the *overlapping* problem, we emphasize in Table 3.4 the negative effect of filtering, which prevents the machine learning model from compensating for this incorrect prior bias. We highlight that such effect also happens even when the parameter values of the (misspecified) first-principles model are perfectly known.

Limitations and future perspectives While our theoretical developments have been elaborated in the context of static supervised regression, they unfortunately do not hold when moving to dynamical systems forecasting problems. Indeed, as the temporal aspect is introduced in the problem definition, it adds a layer of complexity which no longer falls into the predefined set of assumptions. Therefore, an extension of the latter set to dynamical systems along with a similar environment of control experiments represent an interesting future direction to explore, in order to characterize the behavior of hybrid additive training methods regarding their ability to retrieve physically plausible estimations.

7.2 STRAYING FROM THE BEATEN TRACKS OF HYBRID ADDITIVE MODELING

In Section 2.1.2, we mention some seminal works which paved the way for the design of modern hybrid additive training approaches. A remarkable common point for the vast majority of both classical and modern hybrid models is their reliance on the use of neural networks as machine learning component. Indeed, apart from certain exceptions where tree-based models have been combined with linear priors (J. Chen et al., 2007; Zhang et al., 2019), neural networks remain the go-to model for such residual modeling tasks, with more or less modern architectures, likely due to their representative power and their practical flexibility of training. Furthermore, in order to enhance the physical meaning of the first-principles model, recent hybrid additive approaches introduce intricate regularization schemes that control the contributions of the model-based component and of the statistical component. Such process typically induces increased computation times or, at the very least, they burden the thought process as practitioners need to carefully decide on the set of regularization hyperparameters to be tested. In regard to such observations, we have decided to diverge from implicit traditions in a dual manner.

Firstly, with prior knowledge defined as a partial (incomplete) parametric model, we design a novel hybrid modeling training approach which alleviates the need for regularization. In broader terms, our training approach fully decouples the learning of both components, which makes its analysis more straightforward. By leveraging the partial dependence estimations of a machine learning model, we are able to derive an additional dataset for the training of the first-principles model. In terms of theoretical guarantees, we show that our training method is less restrictive as it does not require A3, hence it remains perfectly valid with correlated features. On a range of static regression problems, we show that while it yields models that are as well-performing as other hybrid training methods, its estimation of the physical model is in general much more faithful to the true model. In particular, while all tested hybrid methods failed to recover the correct sign for the slope parameter in the *correlated linear* problem under normal conditions (unfiltered inputs), our PD-based approach did not struggle. Under similar conditions of unfiltered inputs, we show on the *correlated Friedman* problem that correlated features are indeed not problematic to recover faithful prior estimates. We also extend our approach to the problem of dynamical systems forecasting and evaluate it on several systems which have been studied in previous works (Yin, Le Guen, et al., 2021; Donà et al., 2022). As highlighted in 4.2, we show that it remains competitive while being simpler to optimize, hyperparameter-wise. Furthermore, it also provides more faithful prior estimations as can be seen in Tables 4.1 and 4.2.

Secondly, inspired by Grinsztajn et al. (2022) which emphasizes the superiority of tree-based models over neural networks in the context of tabular data, we have decided to investigate the use of those traditional ML approaches as statistical component of hybrid additive models. We show that most of the hybrid training methods can handle such types of model but more importantly that their performance can be competitive with neural-network-based approaches. Indeed, although their predictive performance often lags behind, the physical faithfulness of the obtained models is either on par or much better than that of the latter methods. For instance, on both the *Friedman* and *correlated Friedman* problems, we show that both gradient boosting (Friedman, 2002) and random forests (Breiman, 2001) yield the best first-principles estimation, as can be observed in Table 3.1 and Table 3.3. We also highlight the limitations of tree-based methods. For instance, on the *correlated linear* problem, we notice that PD-based training with neural networks yields a more faithful estimation of the target affine function, which can be explained by the fact that tree-based models struggle to fit affine shapes, therefore the

corresponding PD estimation could be less smooth. Furthermore, trees have, by definition of their learning algorithm, a weak capacity of extrapolation outside the training domain, as opposed to neural networks which can provide a smooth approximation outside the domain, although not necessarily more accurate. Still, we have shown on two real-world problems that introducing a simple linear prior as additive bias can greatly improve model performance in extrapolation settings, as observed in Table 3.6. Finally, traditional tree-based models such as those used in this thesis are non-differentiable as they use axis-aligned splits. Therefore, they cannot be easily extended to address the problem of dynamical systems forecasting.

Limitations and future perspectives The main limitation of our PD-based approach is that the ML model must provide a decent representation of the true input-output relationship, in order to provide useful information to fit the physical model. For instance, if the training sample size is too small, then the ML model could yield a PD estimation of poor quality, as can be observed in Figure 3.1. In fact, even though the training of the physical component and of the final ML model are decoupled, there still exists some level of interaction between both types of models through the iterative fitting of the first-principles model on the successive residual PD estimations of the ML model. Therefore, it is hard to discern a priori cases which would lead to failure, in the sense of physical faithfulness, from cases which could be recovered. It would thus be interesting to study in even more depth the behavior of PD-based hybrid optimization under controlled failure experiments.

Regarding hybrid modeling with tree-based methods, as has already been mentioned, traditional boosting or random forests methods lack the property of differentiability, which truly complicates their direct application into frameworks such as Neural ODE (R. T. Q. Chen et al., 2018), which would allow straightforward integration to unroll predicted trajectories. On the one hand, it could be possible to design a simple working solution with both methods if we enforce the use of the Euler integration scheme. While predictive performance is likely expected to decrease, it would still be interesting to assess the physical plausibility of the resulting prior models as they have shown promising results on that matter in static supervised regression. On the other hand, there exist tree-based alternatives which resolve the problem of differentiability (Irsoy et al., 2012; Frosst et al., 2017; Yang et al., 2018; Mao et al., 2026) and that could, in principle, be integrated within differentiable solvers. Experimenting with such architectures definitely represents a step forward as it also lightens interpretability issues that are inherent to neural networks in general.

As concerns time consumption, PD-based modeling is certainly not ideal in opposition to methods such as APHYNITY which jointly optimize both components in a single training pass. Yet, although being a limitation by construction of the algorithm, it should rather be seen as a trade-off whose importance depends on the pursued objective. If one is looking for faithful prior estimates with no specific time budget, then PD-based training should most surely be considered. However, if one rather wants to maximize final predictive performance while minimizing training time, more traditional approaches are likely a better fit. In any case, it would be insightful to rigorously evaluate training times for the hybrid approaches considered in this thesis. Moreover, it would also be interesting to derive a comparison of model performance for PD-based optimization when sub-sampling data for the training of the iterative ML model, or when varying the number of training epochs for the latter.

Finally, we have quickly covered the topic of hybrid modeling under data scarcity conditions. On the *correlated Friedman* problem, we have shown in Figures 3.1, 3.2 and 3.3 that hybrid additive models usually outperform standard data-driven models, for all training sizes. Yet, the comparison and discussion remain quite limited, as we have only

discussed results for a single experiment. Furthermore, our generic observation should be nuanced. Indeed, even though the performance gap between hybrid and data-driven models seems to follow an increasing trend when reducing training sample size, it seems that for sufficiently low sample sizes (30 samples), this gap starts reducing again and we can even observe in Figure 3.1 that PD-based optimization with MLPs are worse than MLPs alone. Hence, a deeper investigation of the impact of training size and a more thorough study of tree-based methods against neural networks on that matter would be informative.

7.3 SOFTENING THE ADDITIVITY BIAS THROUGH EXPLANATION-GUIDED LEARNING

Broadly speaking, the structural modeling assumption of previously discussed hybrid additive methods may appear too restrictive, as it fundamentally limits the representation space of such hybrid models. One could always argue that the statistical component of these hybrid models could capture arbitrarily complex signals, yet the combination of both models would remain additive. For regularized versions such as APHYNITY or alternate training, the goal is to maximize the contribution of the physical prior to the total output, which prevents the prior from disappearing. For our PD-based optimization algorithm, although non-regularized, the training process is still decoupled and the ML model truly learns an additive residual. However, one can easily imagine problems for which the additivity bias would no longer hold, as for the simple product problem introduced in Section 5.4. Furthermore, it may be difficult, without any loss of generality, to impose *a priori* a particular form on the combination of both models for the targeted input-output relationship. For instance, we may know that the problem is not additive for a function of given inputs, but it may certainly be a more complex and non-linear combination than a simple product, even if some partial parametric model of these inputs is known beforehand.

With this in mind, we have designed a training method which integrates prior knowledge in the same form as hybrid additive models, i.e., as a parametric function of a subset of input features $\mathbf{x}_k$, while being less restrictive on the modeling assumptions. Specifically, by assuming only knowledge about the expected partial dependence of a model on given inputs, we only require to regularize the PD of a single ML model. Conceptually, this moves the methodology from traditional physics-informed modeling to explanation-guided learning, a field whose goal is to improve model performance by ensuring, during training, that its explanations (or interpretations) align with given expert knowledge. Therefore, there no longer exists any (additive) restriction on the combinations of functions of the inputs of interest with functions of the other inputs. Explanation-guided learning can be further motivated by the fact that, if some knowledge about model explanations is available *a priori*, then it should be easy, *to some extent*, to model such form of knowledge.

On a range of synthetic and real-world regression problems, including dynamical systems forecasting problems, we show how such training methods improve both model performance as well as its alignment to prior knowledge about the true partial dependence function. In more details, we study several configurations of PD knowledge, each making distinct assumptions on the level of knowledge and on the practical enforcement of the constraint. In comparison to unconstrained neural networks, we show that PD-constrained models usually perform better both in terms of predictive performance as in the quality of the resulting PD. As can notably be seen in Figure 5.1, PD-constrained methods leveraging perfect knowledge, i.e., up to the values of the PD parameters, generally outperform PD-constrained models which rely on approximated

PD function, i.e., for which the parameters of the PD had to be estimated.

Yet, this observation needs to be nuanced. Indeed, still in Figure 5.1, we can observe that for sufficiently small training sample sizes (50), *linspace* constraining, i.e., constraining on evenly-spaced points (even with approximated PD functions) yield better-performing models with a more faithful estimation of the PD than models constrained only on points sampled from the data distribution (*in-distribution* constraining). We also show that there exist problems for which the former constraining systematically outperforms the latter, as underlined in Figure 5.12 (left), even with worse PD estimates (right). This can be explained by the following reasons. Firstly, the prior approximation of PD parameters must be decent enough, otherwise the model would be constrained with poor-quality data and most likely degraded. Secondly, *linspace* constraining helps the model to have a better global representation of its PD on a wider span of the input domain, as opposed to *in-distribution* constraining which acts more locally. Hence, for such points $\mathbf{x}_k$ which fall outside the training domain, we enforce the model to align with the target PD, which can be really beneficial for problems with correlated input features. Indeed, it is known that PD estimations extrapolate badly in unknown regions of the input domain through the creation of artificial samples, as the model output for such samples is undefined. Therefore, our *linspace* constraining approach helps to address this issue. In fact, we could argue that PD constraining is a step forward to improve the causality of the model. Given that partial dependence represents the average response of a model for certain values of the features of interest $\mathbf{x}_k$, then by constraining the PD on ideal knowledge, we could expect interventions on $\mathbf{x}_k$ to be directly interpretable, on average. This positive impact seems amplified for *linspace* constraining, yet as we did not truly measure such behavior, we cannot provide guarantees on this causality matter.

We also underline the positive impact of PD-constraining methods on model performance under different challenging conditions, namely data scarcity and change of distribution. Indeed, for all experiments, we show that PD-constrained models are much more robust when facing limited training data compared to unconstrained neural networks. As was already evoked for hybrid additive models, introducing prior knowledge into the learning process of ML models improves data efficiency. Furthermore, the different experiments simulating distribution shifts through extrapolation settings on $\mathbf{x}_k$ suggest that *linspace* constraining can be a great solution to dampen these effects in such scenarios.

As the form of prior knowledge is identical to hybrid additive models but the training algorithm (and modeling assumptions) different, it was interesting to compare both methodologies to try to measure the impact of relaxing the inductive bias. For additive problems, we could expect the PD of APHYNITY and PD-based optimization to be correctly modeled as the true PD is actually additively combined, but speculations on more complex problems are not as straightforward. We verify the former by comparing all methods on the *Friedman* problem, as observed in Figure 5.20. However, we show that for problems which cannot be additively decomposed, hybrid models completely fail, as can be seen in Figure 5.22. It also highlights that such problems structures are naturally handled by PD-constraining approaches.

Lastly, as concerns the prior knowledge quality, we show that with more generic knowledge (e.g., assuming only a periodic function for the *Friedman* problem) or even with misspecified PD functions (e.g., assuming a square-root relationship rather than a logarithmic PD for the *concrete* problem), PD-constrained training remains an interesting approach to regularize ML models, as can be observed in Figure 5.30 and Figure 5.33. This is especially true in extrapolation settings, when using *linspace* constraining.

Limitations and future perspectives A central limitation of PD-constraining methods is that they require the definition of parametric priors for the target PD

function. Indeed, although being less restrictive than hybrid additive modeling assumptions, one could easily question the availability of prior PD functions to be a too strong hypothesis. To get around this limitation, we could imagine different solutions which would provide an estimation of the PD function with no prior parameterization. Firstly, we could draw inspiration from PD-based optimization and estimate the PD from a ML model that would be fit on training data. However, this questions the quality of the target PD, especially in small-data regimes where we noticed that the resulting PD of unconstrained models could be of very poor quality. Secondly, a recent work introduces a method to estimate partial dependence curves directly from training data, avoiding potential representation biases inherited from the model when estimating its PD (Parr et al., 2021). Nevertheless, this method estimates accumulated local effects (ALE) rather than true PD effects, and regularizing ALE is much less straightforward than regularizing PD effects, in particular when the size of $\mathbf{x}_k$ is larger than one. Finally, we could also imagine a modeling pipeline where we would first run a symbolic regression method on the training data to recover a first guess of analytical input-output relationship. We could then derive the resulting PD target function, and apply PD-constraining methods accordingly. Nonetheless, above all and maybe more philosophically, all these solutions would slightly contradict the concept of introducing prior knowledge, as knowledge would no longer be assumed to be known but would rather need to be estimated from empirical observations.

As concerns the failure cases of PD-constraining methods, we hypothesize that constraining with poor priors most likely degrades model performance, or at the very least limit, in a negative manner, its representation capacity. Although we have shown with two experiments that approximate knowledge could still be beneficial as constraining information, we did not conduct an extensive study of the behavior of the method under more diverse misspecification conditions. As we have only implemented prior PD fitting on the raw outputs, we could expect our method to fail on the previously discussed *correlated linear* problem, as the estimated slope would have the opposite sign, but it might as well recover during training and thus needs to be explored. It would also be interesting to apply PD-based constraining with prior parametric knowledge that does not entirely represent the true PD but which is still structurally correct in regard to the true input-output relationship. This could for example be studied with aforementioned *overlapping* problem structures, as the one defined in Section 3.4. We also only limited our extrapolation experiments to the extrapolation of a single feature in $\mathbf{x}_k$, yet a deeper study of extrapolating conditions, e.g., extrapolating $\mathbf{x}_{-k}$, would provide supplementary insightful observations on the behavior of the resulting models.

While the previous hybrid PD-based optimization alleviated the need for complex regularization of the first-principles model, PD-constraining methods introduce a regularization balance between the mean-squared error minimization and the minimization of the PD distance. For the sake of simplicity, we have decided to optimize an affine combination of both criteria, which could however limit the power of the method. An interesting direction of future work would be to elaborate on variants of PD-constraining which would greatly simplify the optimization of regularization hyperparameters.

A light limitation of PD-constraining methods is the amount of computation time, which grows exponentially with the size of $\mathbf{x}_k$, if we wanted to evaluate the PD for every value of $\mathbf{x}_k$. Still, we do not expect this to be a practical problem as PD-constraining is inherently linked to the concept of explanations, which implies that $\mathbf{x}_k$ would most likely always have a size smaller or equal to two, for PD visualization purposes. Furthermore, as is already implemented in the experiments of Chapter 5, it always remains possible to sub-sample random values from $\mathbf{x}_k$ for each mini-batch, during training.

Finally, a future work direction could be to try to extend PD-constraining methods to other types of regression models, such as traditional tree-based models. Although this extension would not be as straightforward as for neural networks, they remain a desirable ML solution as they are naturally more interpretable and easier to optimize. As was the case for hybrid additive models, we could consider differentiable variants (Irsoy et al., 2012; Mao et al., 2026). Another extension could consider individual conditional expectation (ICE) measures, an interpretation technique which describes the model output of a given sample, when varying only the features of interest $\mathbf{x}_k$. Using ICE knowledge would allow the modeling of heterogeneous effects and explanations, which is not possible with PD-constraining due to the averaging. Still, such ICE-constraining methods would likely require the definition of a parametric prior function.

7.4 CRAFTING KNOWLEDGE FROM DATA

As noted before, PD-based optimization and PD-constraining methods both make the assumption of having access to a (partial) parametric model representing the only source of knowledge. Yet, although the existence of this form of knowledge is sensible, either as a sub-model (for PD-based optimization) or as a target explanation (for PD-constraining methods), its availability remains questionable. As has already been discussed earlier in this Chapter, non-parametric variants of similar knowledge would be considered more practicable. For these reasons, we have decided to study an even less restrictive integration of prior knowledge, which rather assumes that the essence of knowledge is entirely contained into data samples, through pairs of features of interest $\mathbf{x}_k$ and their associated output. In fact, while this source of knowledge is more easily accessible, we highlight that the best least-squares ML model fit on such pairs of data asymptotically approaches the true marginal plot (or M-plot) w.r.t. $\mathbf{x}_k$, i.e., the average true response at those values. Hence, an approximation of the M-plot could serve as a form of prior knowledge.

In the practical context of learning with small data, we have designed a novel modeling framework as well as its associated training algorithm which leverages this M-plot approximation in order to dampen the impact of data scarcity. In particular, by modulating the learned M-plot estimate with both scaling and shifting functions and by penalizing deviations of the modulated model from the M-plot estimate, we show on a range of real-world tabular regression problems that the resulting model achieves better predictive performance than traditional regularized versions of neural networks, as can be observed in Figure 6.2. We motivate the approach in regard to the explainable AI (XAI) literature and provide theoretical elements to support the soundness of the method. Specifically, we show that the M-plot of our regularized loss function converges towards the target M-plot estimate. Moreover, we compare the predictive performance of M-plot modulation against several related baselines and show that it usually achieves more desirable results, although not always by a significant margin. Finally, we highlight that our modulated learning method with M-plot regularization is usually the most robust under very scarce data conditions (50 training samples), while the gap reduces when training sample size increases.

To deepen the exploration of the integration of such type of non-parametric (approximate) knowledge, we design an ablation study to disentangle the impact of the modulation from that of the M-plot regularization. In Figure 6.3, we show that regularizing any modulation (let it be only additive or also multiplicative) generally helps to improve final performance, while the optimal choice of modulation seems to be very dependent on the underlying problem structure. In particular, we show that for the *product* problem, the scaling modulation has a truly beneficial impact, whereas it appears less clear for other problems.

Limitations and future perspectives Although conceptually appealing, the M-plot modulating approach comes with several limitations. To dampen the impact of limited training data, the method provides a very decent starting point, yet it remains highly dependent on the quality of the M-plot itself. Indeed, a poor estimation of the M-plot would most likely make learning much more complex as both the scaling and shifting networks would need to compensate for modeling errors. Additionally, regularizing the model with too noisy estimates would also restrict even more its representation capacity. More importantly, the method does not come with any optimality guarantee about model performance, aside from the conservation of the final M-plot. Actually, this is implied by the lack of prior assumptions on the final model, e.g., we cannot truly expect better M-plot performance as it is unknown. Even if the latter was known algebraically, it still remains tricky to measure the M-plot faithfully (Molnar, 2020), hence it is hard to judge on its quality as an explanation. Finally, we did not investigate failure cases of the method. An easy way to quantify its sensitivity would be to experiment with varying sizes for the large “knowledge” database to try to establish a threshold on the minimum number of training samples to obtain reasonable M-plot estimates, although it might be very problem-dependent. It would also be interesting to explore other synthetic problem structures to derive some sort of prior conditions on the problem which, if verified, would increase the chances of success of the method in limiting overfitting.

More generally, in terms of empirical validation of the regularized modulated technique, several elements should be added to give more scientific weight to the method. There exist additional baselines to which the modulated approach could be compared. For instance, we could simply fine-tune the M-plot approximation on the small database, by replacing the input layer. However, in such a setting, one would probably lose the interpretable trait of the original M-plot estimate. Another possible baseline is to modulate the M-plot estimate with scaling and shifting functions that do not take $\mathbf{x}_k$ as input. While conceptually very close to the proposed approach, its predictive performance may be more limited if the M-plot estimate is kept frozen as neither scaling nor shifting would be able to compensate for errors contained in the M-plot. Nevertheless, this variant would preserve the M-plot by construction and therefore not require any regularization.

Moreover, the current study of the modulated approach remains limited to a single choice of $\mathbf{x}_k$, i.e., some of the most important input features. While it allows getting an idea about the potential of the method, it remains arbitrary and it would be interesting to study the impact of the importance of the selected variables on the performance of the method (e.g., selecting random features) as it would also be interesting to investigate the effects of varying the size of $\mathbf{x}_k$. Also, the current study lacks an application of the method in real testing conditions, where one would truly have access to a larger database of easily measurable covariates.

Lastly, as was the case for PD-based optimization and PD-constraining methods, an extension to more traditional ML methods such as tree-based models definitely represents an interesting line of future work. For instance, we could imagine a setting where the M-plot estimate is represented with neural networks, while shifting and scaling functions are modeled by tree-based methods (and vice-versa). Another possibility is to apply the technique entirely with tree-based models, and to compare the performance of the latter against their neural-network-based counterparts. In any case, the technique gives insightful directions on future techniques to experiment with this type of modulation. For example, we could consider a setting closer to FiLM methods (Perez et al., 2018), where $\mathbf{x}_k$ (or $\mathbf{x}_{-k}$, depending on the setting) would represent some context information. It could also be interesting to explore the application of such techniques to the problem of dynamical systems forecasting, but also with other types of modalities than pure tabular data. Indeed, it is easy to picture a classification problem based on images which would be enriched with textual or numeric data.

7.5 KNOWLEDGE-GUIDED MACHINE LEARNING: REFLECTIONS AND FUTURE DIRECTIONS

Taking a step back from our previously discussed contributions and their related limitations, we highlight in the following some future directions or limitations of current works in the field of knowledge-guided machine learning that we deem interesting.

7.5.1 MODEL EVALUATION

Broadly speaking, we believe that there exists a true limitation in the fair evaluation of hybrid methods, as concerns their predictive performance in practice. This observation certainly applies to this thesis, whose application to real-world problems remains quite limited in general. Still, restricting at least to the context of recent works on hybrid additive models (e.g., APHYNITY (Yin, Le Guen, et al., 2021) or alternate optimization (Donà et al., 2022)) and looking beyond the issue of reproducibility (i.e., hard-to-reuse or non-existing implementations of the training methods which hinder fair assessment), there lacks above all an effort to compare methods on a common experimental basis. Although methods are compared on similar experiments, e.g., PDE systems, eventually the trajectories generated to simulate such systems stem from different implementations, which makes the final evaluation less robust. Regarding synthetic experiments, the recent works of Takamoto et al. (2022) and Ohana et al. (2024) introduce a suite of datasets covering diverse spatiotemporal PDE systems with access to the respective PDE parameters, making them suitable for both forward and inverse problems. Furthermore, Gamella et al. (2025) developed two *causal chambers*, which allow the evaluation of models on real-world experimental data obtained from controlled environments, a very desirable property to assess the robustness of models in practical conditions. These works thus represent solid benchmarks on which future knowledge-guided methods should be compared, depending on the type of prior knowledge.

7.5.2 REGULARIZATION

As was already evoked in Chapter 3, current hybrid additive modeling approaches involve intricate regularization procedures to properly estimate the contribution (and parameters) of the first-principles model while maintaining predictive performance, with hyper-parameter ranges that may vary depending on the class or instance of the considered problem whereas the nature of the regularizer depends on the class of the chosen statistical model. We argue that PD-based hybrid optimization takes a step in the direction of hybrid methods that alleviate heavy regularization procedures, although the method itself presents some (mentioned) caveats such as increased computing times linked to PD estimations or simply its inherent restriction to additive problems. We believe this disentanglement from model-specific regularizers to be of utmost importance for the development of flexible hybrid methods, encouraging the implementation of model-agnostic approaches, e.g., with applications to more traditional machine learning models such as tree-based methods. A notable example is the recent work of Takeishi (2026) where a model-agnostic hybrid training approach is implemented through the concept of sharpness-aware minimization. Even though this method still requires some level of regularization, the nature of the regularizer itself does not depend on the model, it relies on a single hyperparameter and covers non-additive settings as well. More globally, it is important to try to develop hybrid methods that are model-agnostic by construction, in the same philosophy as post-hoc interpretation techniques.

7.5.3 GENERALIZATION AND TRANSFERABILITY

Given the recent shift toward machine learning research on large language models and their practical surge, one could obviously question the use of similar foundation models for more structured data, as can be found in the form of tabular data. Current implementations of such instances, including TabPFN (Hollmann et al., 2022) and the more recent TabFM (Kong et al., 2026), are trained entirely on synthetic data generated from structural causal models and can yield predictions for a given real-world dataset in a single pass. Foundation models typically take value from the fact that they are very easily transferrable across tasks, hence we could be tempted to exclusively rely on such generic models. In the context of low-data regimes with prior knowledge of various forms, it could be interesting to study the behavior and robustness of foundation models which could be fed with additional data including, e.g., (known) predicted residuals, or to directly use the foundation model to predict the residual based on some prior approximate model. In particular, it would be of general interest to assess whether more traditional grey-box modeling approaches (such as hybrid additive models) perform better than tabular foundation models in terms of predictive performance.

To follow on the concepts of model generalization and model transferability, one may argue about the practical efficiency and relevance of the hybrid additive modeling frameworks presented throughout this thesis. Indeed, when facing distribution shifts or evolving conditions from the initial training environment, such models appear more limited as, in their simplest forms, they are confined to representing a single instance of a given process or dynamical system and hence cannot naturally adapt to test-time shifts. Furthermore, although there also exist foundation models specifically designed for time-series forecasting, e.g., TimesFM (Das et al., 2023), they remain very rigid and cannot easily address data drifts as they rely entirely on their pre-training (Soni et al., 2026). We believe domain-adaptive architectures to be of great importance for the field of knowledge-guided machine learning. Several works already implement some principles to improve domain adaptation, for example assuming additive adaptation (either data-driven (Yin, Ayed, et al., 2021) or through a shared physical prior (Osaka et al., 2025)), or relying on (sparse) hyper-networks taking as input environment-specific features (Kirchmeyer et al., 2022; Kassai Koupai et al., 2024; Brenner et al., 2025) or, more recently, leveraging diffusion processes to adapt model weights in latent space (Li et al., 2025).

7.5.4 MODEL INTERPRETABILITY

A final important subject is the topic of model explainability, as it remains a major concern for the future development and deployment of novel machine learning models in practice. For example, although not strictly required under precisely-defined forms of interpretability, the European Union AI Act requires systems to be *transparent* enough such that their decision process can be easily analyzed (European Parliament et al., 2024). This is particularly relevant for critical domains such as the healthcare ecosystem where models should treat confidential information whilst avoiding ethical biases, making practitioners more skeptical and reluctant about their practical integration. Hence, there remains much effort to accomplish as regards model interpretability in general. As was mentioned in Section 7.3, prior domain knowledge regarding model interpretability should be easy to derive in the first place, thereby making explanation-guided learning one of the most important research directions in order to bridge the gap toward adoption by domain practitioners.

On the one hand, as concerns feature importance, such property can be difficult to quantify a priori by domain experts. Although early works in explanation-guided learning focused on favoring (or penalizing) certain attributions of features and interactions (Rieger et al., 2020), a few recent methods focused on similar guidance techniques leveraging

more generic properties such as the relative importance of feature (Erion et al., 2021), but also representing more general concepts (Balayan et al., 2020) or even introducing meta-features which contain information on why certain features are more important than others (Weinberger et al., 2020). Therefore, we believe that a promising future direction concerns the expansion of such notions of concepts and meta-features and how they relate to raw tabular features, in a tentative to improve the trust in the resulting ML models. Indeed, prior information of this type should be more easily designable by domain experts compared to fixed attribution priors on raw feature columns.

On the other hand, regarding model response to particular features, we believe that research on improving model faithfulness given prior functional knowledge is currently under-explored. PD-constraining methods presented in Chapter 5 therefore represent a step paving the way for further developments. Naturally, the main question concerns the acquisition of such forms of prior knowledge in the first place, especially for problems whose causal relationships between inputs and output are mostly unknown. Yet, an interesting future work would consider similar steering techniques applied with more generic knowledge, e.g., at the scale of properties such as monotonicity, but also with groups of data which are known to respond differently according to some features. By relying on ICE-constraining (as opposed to PD-constraining), it would be possible to model such heterogeneous effects.

Finally, making models more interpretable by construction represents a major milestone, even though current attempts seem quite limited as they usually cover poorly-expressive models such as linear regression or rule-based methods. Incentives such as self-explaining neural networks (S. Lee et al., 2022) are a great example showing how to incorporate reasoning about the explanations directly into the decision (and training) process of the models. Through their antecedent generative module, the model becomes directly interpretable, yet it remains quite restrictive as it can only sample a single rule to make its final decision, but it certainly opens paths for additional creative solutions.

7.6 CONCLUSION

As a matter of general conclusion, this thesis emphasizes the benefit of introducing prior knowledge into the training process of machine learning models. Instead of solely relying on either rigid physical models or on more flexible data-driven models, it calls for a solution at the intersection of both modeling frameworks. Through different levels of integration of various forms of domain knowledge, we show how the resulting hybrid models can be more data efficient, more interpretable and, ultimately, more robust.

REFERENCES

- Balayan, Vladimir et al. (2020). “Teaching the machine to explain itself using domain knowledge”. In *arXiv preprint arXiv:2012.01932*.
- Breiman, Leo (2001). “Random forests”. In *Machine learning* 45.1, pp. 5–32.
- Brenner, Manuel et al. (2025). “Learning interpretable hierarchical dynamical systems models from time series data”. In *International Conference on Learning Representations*. Vol. 2025, pp. 85616–85652.
- Chen, Jinbo et al. (2007). “A partially linear tree-based regression model for assessing complex joint gene–gene and gene–environment effects”. In *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* 31.3, pp. 238–251.
- Chen, Ricky T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In *Advances in Neural Information Processing Systems*.
- Das, Abhimanyu et al. (2023). “A decoder-only foundation model for time-series forecasting”. In *arXiv preprint arXiv:2310.10688*.
- Donà, Jérémie et al. (2022). “Constrained physical-statistics models for dynamical system identification and prediction”. In *ICLR 2022-The Tenth International Conference on Learning Representations*.
- Duarte, Belmiro, PM Saraiva, and CC Pantelides (2004). “Combined mechanistic and empirical modelling”. In *International Journal of Chemical Reactor Engineering* 2.1, pp. 1542–6580.
- Erion, Gabriel et al. (2021). “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”. In *Nature machine intelligence* 3.7, pp. 620–631.
- European Parliament and Council of the European Union (2024). “Regulation (EU) 2024/1689...”. In *Official Journal of the European Union* L 1689.
- Feynman, Richard Phillips (1965). “The Character of Physical Law”. MIT Press.
- Friedman, Jerome H (2002). “Stochastic gradient boosting”. In *Computational statistics & data analysis* 38.4, pp. 367–378.
- Friedman, Jerome H, Eric Grosse, and Werner Stuetzle (1983). “Multidimensional additive spline approximation”. In *SIAM Journal on Scientific and Statistical Computing* 4.2, pp. 291–301.
- Frosst, Nicholas and Geoffrey Hinton (2017). “Distilling a neural network into a soft decision tree”. In *arXiv preprint arXiv:1711.09784*.
- Gamella, Juan L, Jonas Peters, and Peter Bühlmann (2025). “Causal chambers as a real-world physical testbed for AI methodology”. In *Nature Machine Intelligence* 7.1, pp. 107–118.
- Grinsztajn, Léo, Edouard Oyallon, and Gaël Varoquaux (2022). “Why do tree-based models still outperform deep learning on typical tabular data?” In *Advances in Neural Information Processing Systems* 35, pp. 507–520.
- Hollmann, Noah et al. (2022). “TabPFN: A transformer that solves small tabular classification problems in a second”. In *arXiv preprint arXiv:2207.01848*.
- Irsoy, Ozan, Olcay Taner Yıldız, and Ethem Alpaydın (2012). “Soft decision trees”. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*. IEEE, pp. 1819–1822.

- Kassai Koupai, Armand et al. (2024). “Boosting generalization in parametric pde neural solvers through adaptive conditioning”. In *Advances in Neural Information Processing Systems* 37, pp. 70659–70692.
- Kirchmeyer, Matthieu et al. (2022). “Generalizing to new physical systems via context-informed dynamics model”. In *International conference on machine learning*. PMLR, pp. 11283–11301.
- Kong, Weihao and Abhimanyu Das (2026). “Introducing TabFM: A zero-shot foundation model for tabular data”. In *Google Research Blog* 30.
- Kramer, Mark A, Michael L Thompson, and Phiroz M Bhagat (1992). “Embedding theoretical models in neural networks”. In *1992 American Control Conference*. IEEE, pp. 475–479.
- Lee, Dae Sung, Peter A Vanrolleghem, and Jong Moon Park (2005). “Parallel hybrid modeling methods for a full-scale cokes wastewater treatment plant”. In *Journal of Biotechnology* 115.3, pp. 317–328.
- Lee, Seungeon et al. (2022). “Self-explaining deep models with logic rule reasoning”. In *Advances in neural information processing systems* 35, pp. 3203–3216.
- Li, Ruikun et al. (2025). “Generative Adaptation of Dynamics to Environmental Shifts via Weight-space Diffusion”. In *arXiv preprint arXiv:2505.13919*.
- Mao, Qiangqiang et al. (2026). “Differentiable Decision Tree via” ReLU+ Argmin” Reformulation”. In *Advances in Neural Information Processing Systems* 38, pp. 166874–166909.
- Molnar, Christoph (2020). “Interpretable machine learning”. Lulu. com.
- Ohana, Ruben et al. (2024). “The well: a large-scale collection of diverse physics simulations for machine learning”. In *Advances in Neural Information Processing Systems* 37, pp. 44989–45037.
- Osaka, Akira, Naoya Takeishi, and Takehisa Yairi (2025). “Domain Adaptation with Hybrid Modeling for Learning Dynamical Systems”. In *The 17th Asian Conference on Machine Learning (Conference Track)*.
- Parr, Terence and James D Wilson (2021). “Partial dependence through stratification”. In *Machine Learning with Applications* 6, p. 100146.
- Perez, Ethan et al. (2018). “Film: Visual reasoning with a general conditioning layer”. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32 (1).
- Rieger, Laura et al. (2020). “Interpretations are useful: penalizing explanations to align neural networks with prior knowledge”. In *International conference on machine learning*. PMLR, pp. 8116–8126.
- Soni, Kavin, Debanshu Das, and Vamshi Guduguntla (2026). “Assessing the Operational Viability of Foundation Models for Time Series Forecasting”. In *arXiv preprint arXiv:2605.24381*.
- Su, Hong-Te et al. (1993). “Integrating neural networks with first principles models for dynamic modeling”. In *Dynamics and control of chemical reactors, distillation columns and batch processes*. Elsevier, pp. 327–332.
- Takamoto, Makoto et al. (2022). “Pdebench: An extensive benchmark for scientific machine learning”. In *Advances in neural information processing systems* 35, pp. 1596–1611.
- Takeishi, Naoya (2026). “Sharpness-Aware Hybrid Model Learning for Architecture-Agnostic Parameter Estimation”. In *arXiv preprint arXiv:2602.06837*.
- Thompson, Michael L and Mark A Kramer (1994). “Modeling chemical processes using prior knowledge and neural networks”. In *AIChE Journal* 40.8, pp. 1328–1340.

- Weinberger, Ethan, Joseph Janizek, and Su-In Lee (2020). “Learning deep attribution priors based on prior knowledge”. In *Advances in Neural Information Processing Systems* 33, pp. 14034–14045.
- Yang, Yongxin, Irene Garcia Morillo, and Timothy M Hospedales (2018). “Deep neural decision trees”. In *arXiv preprint arXiv:1806.06988*.
- Yin, Yuan, Ibrahim Ayed, et al. (2021). “LEADS: Learning dynamical systems that generalize across environments”. In *Advances in Neural Information Processing Systems* 34, pp. 7561–7573.
- Yin, Yuan, Vincent Le Guen, et al. (2021). “Augmenting physical models with deep networks for complex dynamics forecasting”. In *Journal of Statistical Mechanics: Theory and Experiment* 2021.12, p. 124012.
- Zhang, Haozhe, Dan Nettleton, and Zhengyuan Zhu (2019). “Regression-enhanced random forests”. In *arXiv preprint arXiv:1904.10416*.