Learning simplified functions to understand

Bruno Apolloni[1] and Ernesto Damiani[2]

[1] Dipartimento di Scienze dell'Informaziones Università degli Studi di Milano
apolloni@di.unimi.it
[2] Center on Cyber-Physical Systems, Khalifa University,
ernesto.damiani@ku.ac.ae

Abstract. We propose an unprecedented approach to post-hoc interpretable machine learning. Facing a complex phenomenon, rather than fully capturing its mechanisms through a universal learner, albeit structured in modular building blocks, we train a robust neural network, no matter its complexity, to use as an oracle. Then we approximate its behavior via a linear combination of simple, explicit functions of its input. Simplicity is achieved by (i) marginal functions mapping individual inputs to the network output, (ii) the same consisting of univariate polynomials with a low degree,(iii) a small number of polynomials being involved in the linear combination, whose input is properly granulated. With this contrivance, we handle various real-world learning scenarios arising from expertise and experimental frameworks' composition. They range from cooperative training instances to transfer learning. Concise theoretical considerations and comparative numerical experiments further detail and support the proposed approach .

Keywords: Explainable AI, Post-hoc Intepretable ML,ridge polynomials, compatible explanation, transfer learning, minimum description length.

1 Introduction

Willing to face complex phenomena, we surrendered to the idea that someone provides immediate answers to our questions about them. It is our typical attitude when we query research engines (we google something out) or any URM (universal responder machine), in so reviving old myths such as Delphi Oracle or Golem. The same occurs when we decide to solve either a regression or a classification problem via cognitive algorithms, such as a neural network. The operational pact is: once we assume that one of the above subjects is well assessed, we may inquire, get a response, but we are not allowed to ask why that answer.

Cognitive systems provided formidable solutions to many problems, such as speech to text or text to speech translation. The rapid evolution of deep

neural networks and their pervasion in many appliance interfaces, jointly with the widespread habit to search answer from online URM, prospect disquieting Orwellian scenarios. Whole communities of people renounce understanding why some answers occur, as a modern implementation of the ancient statement "contra factum non valet argumentum" [4], with answers taking the value of facts in this scenario.

Willing to elaborate on complex phenomena such as jam traffic in a crowded city, structural stress of a bridge, or money lending in a FinTech environment, we need stating relationships between the target variable y (such as traffic intensity and stress value) and the broad array $\mathbf{x}$ of input variables characterizing the phenomenon environment. Let us assume having an algorithm g available so that for many $\mathbf{x} \in \mathcal{X}$, $g(\mathbf{x}) \simeq y$ with a satisfactorily approximation. However, g may be not understandable, either because unknown, like an Oracle, or because it is a function so complex and endowed with so many parameters that we cannot perceive its real trend (or any intermediate condition). Lack of understandability is a typical issue for Artificial Neural Networks and raised the search for techniques deducing formal expressions from trained neural networks, a goal loosely recapped in the sentence: "passing from synapses to rules" [3] since the Eighties of the past century. This issue currently revives, being addressed to the AI tools in general.

Today, possibly fostered by the general simplificative mood [4], researchers look for understandable functions $\tilde{g}$ replacing g, a thread that the Explainable AI taxonomy [1] denotes as *Intepretable Machine Learning* with a *global-model-agnostic interpretability* goal. Simplification may happen at various levels and with various strategies in search of *surrogate models*, all pivoting around a few strongholds like:

- Linear relations are the most understandable and identifiable as well. The most elementary ones are just the sum of atomic functions plus a bias.
- To be efficient, the atomic functions must be relatively simple, albeit not necessarily linear in turn, hence multi-derivable and with a small number of arguments, possibly only one, through a projection from the entire $\mathcal{X}$. In the last case, we expect the projections to be *grosso modo* orthogonal to one another.

Within the family of Additive Index Models (AIM), the general model can be written as [19]:

$$\tilde{g}(\mathbf{x}) = \nu + \gamma_1 h_1(\beta_1^T \mathbf{x}) + \ldots + \gamma_k h_k(\beta_k^T \mathbf{x}) + \epsilon \tag{1}$$

where

[4] Aristotle, L. IV, Metaph. c. III

- β_i identifies hyperplane projections rendering h_i ridge functions [5]. The general wisdom is to work with orthogonal projections (limit case $\beta_i^T \mathbf{x} = x_i$) in order to exploit $\mathbf{x}$ information better.
- The shape of h_is is up to us. Common solutions are:
  1. h_i a polynomial, possibly linear, so that the overall expression of the complex function corresponds to the solution of a linear regression problem
  2. h_i a spline function, so that we can infer its shape according to entropic criteria [20, 14]
  3. h_i a neural network, to have many parameters (though well organized to support interpretation), perfectly fitting g (at least theoretically) via $\tilde{g}$ [16]
- the coefficients γ_i are 1 in the original AIM [14], which represent a suitable set of parameters in many models. Worth mentioning that interpreting $\tilde{g}(\mathbf{x}) = \tilde{g}(E[\mathbf{x}|y]) = \tilde{g}(\mu)$ with $\epsilon = 0$ leads to Generalized Additive Models (GAM) [9] (Generalized Linear Models (GLM) for h_i linear).

Identifiability [20] and convergence [14] results are available for the options 2 and 3, where convergence holds for the mean square error function on the pair $(g(\mathbf{x}), \tilde{g}(\mathbf{x}))$ for usual training procedure of $\tilde{g}$ as a whole.

Our approach is rather different. We maintain the above general model(1). We train the function in an ensemble learning model with h_i as in option 1, and we put $\beta_i^T \mathbf{x} = x_i$. In other words, we train the single h_i to replicate the marginal shape of g for the individual x_is. This procedure results in a one-shot training. Then we estimate the linear combination of the h_is in (1). Moreover, in order to make sure that the learned h_is stay simple enough to afford human understanding of the overall expression involving them, we bargain $(g(\mathbf{x}), \tilde{g}(\mathbf{x}))$ Mean Squared Error (MSE) with 1) the number of components of $\mathbf{x}$ involved in each h_i and 2) the degree of polynomials in h_i, both according to the minimum description length criterion, and 3) the accuracy of $\mathbf{x}$, i.e., the size of its vector quantization.

The original cognitive system's role remains crucial in our approach, but we move it to the background. A deeply-trained neural network acts as an oracle that supplies all training examples needing a very accurate inference of the approximating functions. These functions supply a manageable interpretation of what the cognitive system may have learned in a sub-symbolic way. Of course, lay users may find the resulting interpretation hard to follow (making it a *non-immediately actionable explanation* according to [1]). Still, we claim it provides a concrete and trustworthy starting point to users with some technical education wishing to find an explanation of the cognitive system's behavior (when trying to understand complex behavior, there is no "free lunch" [18])

[5] Informally we obtain a ridge function of a vector $\mathbf{x}$ computing a univariate function on the inner product between $\mathbf{x}$ and a constant vector $\mathbf{a}$, representing the *direction*. As a result, the ridge function is constant on the hyperplanes that are orthogonal to the ridge direction.

The paper is structured as follows. Section 2 illustrates the theoretical framework of our approach. Section 3 describes our inference procedure rhat we call Marginal Functions Ensemble (MFE). Section 4 provides numerical experiments contrasted with literature results, while concluding remarks are given in Section 5.

2 Compatible functions

While the spring of modern science has been the willingness to develop mathematical theories to describe/explain the world's workings, we may say that the aim of a somehow *post-modern science* is, more modestly, to face the complexity of the phenomena with *compatible tools*. The classical paradigm is airplane wing: it does the same job as birds' wings but is more straightforward and feasible. Facing complex phenomena such as those mentioned in the Introduction, the scientific community first did rely on probability models as the last frontier before the unexplainable (*hic sunt leones*). It later surrendered to cognitive algorithms (with a transition through fuzzy sets frameworks). The final paradigm is: "It does not matter why, provided it works"[11]. This sentence summarizes the general philosophy of cognitive algorithms whose operation inspired by living organisms, human beings *in primis*, and the most common operational paradigm is " learning from examples."

Actually, in the last three decades, hard competition has been fought between two ways of implementing the paradigm. Besides the sub-symbolic one, represented by the cognitive algorithms having neural networks as the main computational tool, the second way, the symbolic one [11], is represented by the computational learning vein having PAC-learning methods as the main theoretical tools.

- Neural networks are families of general-purpose functions endowed with many parameters, to be adaptable to reproduce any computable function (see [7]).
- PAC-Learning addresses specific classes of functions, described by explicit symbolic expressions, to be satisfactorily approximated by items of specific classes of hypotheses, with the same description facilities.

Hence, in search of compatible tools, the second way would appear the most suitable for us.

PAC stands for "Probably Approximately Correct"; the general scheme is the following. Consider a class of functions C that is hard, to some extent, to identify; for instance, circles dividing positive from negative points on a plane [6]. Then, for a given set S of points separated by an unknown circle c, the goal is to draw, basing on S, a circle, call it h (an *hypothesis*) that is very close t c. We have no ambition of discovering precisely c; instead, we look for an h that works

[6] Do not be misled by the simplicity of the function; try to devise from scratch an algorithm that identifies a circle dividing any set of points which in turn are separable by a circle.

almost as well as c, in the sense that, questioned on a new point $\mathbf{x}$ in the plane $\mathcal{X}$, $h(x) = c(x)$ with high probability (see Figure1). In that, it meets precisely our notion of compatibility.

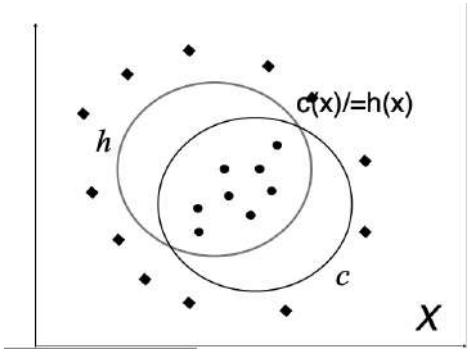

Fig. 1: PAC learning basics. $\mathcal{X} \rightarrow$ the sampling space; $c \rightarrow$ the boolean target function; $h \rightarrow$ the approximating hypothesis. Bullets $\rightarrow$ positive points; rhombuses $\rightarrow$ negative points. Out of the intersection between c and h points will be labeled wrongly by h, i.e $c(x) = h(x)$.

In opposition to the "Turing machine-versus-perceptron" duel for the title of computational paradigm champion, which was won by the former, the Computational Learning paradigm succumbed to neural networks in the learning-by-examples practice. Suppose we know C and H (the class of the hypotheses h), and we can compute a given complexity parameter of their symmetric difference (trivial when of C and H are classes of circles, definitely less trivial in many real-life conditions). In that case, we could determine lower bounds to the learning algorithms' performance as a function of the size of S. This procedure is a remnant of what happens, for instance, expressing a linear regression problem's accuracy. Things, however, are more complex. In the case of boolean functions like labeling circles, we express accuracy in terms of probability of sampling a set S, based on which we may compute an h such that the symmetric difference between c and h has a given probability (hence an error probability since $c(\mathbf{x}) \neq h(\mathbf{x})$ therein) for the $\mathbf{x}$ distribution law. This result appears very powerful since it is distribution-independent (i.e., it holds whatever the distribution law of $\mathbf{x}$), but unfortunately, the upper-bounds constraining the size of S are very loose, outside of any feasible sampling plan. Things work a bit better if we know the distribution law of $\mathbf{x}$, but worse if we do not know C.

Computational Learning Theory still remains a formidable field of theoretical elaborations but has been abandoned for practical purposes. Nevertheless, for the reasons discussed in the Introduction, we believe that the idea of approximating a goal function g with a class H of symbolic functions is a fundamental one to

achieve AI interpretability. In these pages, we try exploiting and adapting results from Computational Learning Theory in this vein.

This style of learning introduces two complexity issues: computational complexity and sample complexity. The former concerns the running time of the algorithm, which computes h. While it proves excessive if H is the class of Disjunctive Normal Forms (DNF) since the related satisfiability problem is NP-complete [8], in our case, where hypotheses consist of linear combinations of a short number of one-dimensional polynomials of small degree, this issue is irrelevant. Sample complexity is appreciated in terms of the upper-bound to the number of samples to observe to have acceptable twin probabilities characterizing the learning goal. This quantity denotes the information amount the learning algorithm needs to identify a hypothesis within its class, thus resulting in a measure of the information richness/difficulty of the function to be learned – a measure that may denote its interpretability by humans.

Let us start from the following theorem as an essential reference for our considerations.

Theorem 1. *[2] For a space $\mathcal{X}$, assume we are given*

- *a concept class C on $\mathcal{X}$ with complexity index $D_{\mathsf{C},\mathsf{C}} = \mu$;*
- *a set $\mathcal{Z}_m$ of labeled samples $\mathbf{z}_m \in \mathcal{X} \times \{0,1\}$;*
- *a fairly strongly surjective function $\mathcal{A} : \mathcal{Z}_m \mapsto \mathsf{H}$ computing consistent hypotheses.*

Consider the families of random sets $\{c \in C : \mathbf{z}_{m+M} = \{(x_i, c(x_i)),\ i = 1, \ldots, m+M\}$ when $\mathbf{z}_m$ spans the the samples in $\mathcal{Z}_m$ and the specifications of their random suffixes $\mathbf{Z}_M$, with $M \to +\infty$ according to any distribution law.

For a given $(\mathbf{z}_m, h)$ and $h = \mathcal{A}(\mathbf{z}_m)$, denote with

- *μ_h the complexity index $\mathrm{D}_{C,H}$ for the adopted computational approximation*
- *t_h be the number of points misclassified by h*
- *$U_{c \div h}$ the random variable given by the probability measure of the simmetric difference $c \div h$between c and h.*

Then, for $m \geqslant \max\left\{\frac{2}{\varepsilon}\log\frac{1}{\delta}, \frac{5.5(\mu_h + t_h - 1)}{\varepsilon}\right\}$, $\mathcal{A}$ is a learning algorithm for every $\mathbf{z}_m$ and $c \in C$ ensuring that $\mathrm{P}(U_{c \div h} \leqslant \varepsilon) \geqslant 1 - \delta$.

Let us highlight the following points:

1. complexity index $D_{\mathsf{A},\mathsf{B}}$ is a function of the symmetric difference between the two classes in argument denoting how clear is their symmetric difference. The most used index is *Vapnik-Chervonenkis dimension* [15], another suitable one is *detail* [5]. The higher the complexity index, the more is the learning difficulty.
2. D increases with the decrease of the computation approximation, in turn, depending on the accuracy.
3. the function $\mathcal{A}$ computing the hypothesis h is consistent if $h(x) = c(x)$ for each $x \in \mathbf{z}_m$. For non-consistent $\mathcal{A}$ we take note of the number t_h of mistaken samples.

4. complexity and approximation of a hypothesis sum up (via $m_h + t_h$) to determine the learning task's difficulty.

Though the theorem refers to classes of Boolean functions, we may derive the main lesson that three strongholds characterize the complexity of the task of learning a function:

1. the complexity of the class C to learn, paired with the class H to get its hypotheses,
2. the approximation of the hypothesis with which we want to explain the labeling of the points,
3. the accuracy with which we register the input data.

When trying to learn real-world functions, sampling complexity plays a crucial role because of the cost of achieving samples. The high and possibly unknown complexity of C, when we abandon case studies, prevents us from hazarding analytical evaluations of the learned function's accuracy, normally left to the toss of test sets. What remains are the directions to improve our learning that find immediate companions in the mechanisms with which our brain may interpret the learned function. Namely:

- complexity of the hypothesis function maps into the number of involved variables and parameters. This mapping represents an operational implementation of *explanation selection* activity [13] that the human brain performs to understand the behavior of an algorithm, or a phenomenon in general, in terms of assignment of causal responsibilities [10]. Up to our experience, the number of variables and parameters should be at most 5. Otherwise, most humans would not be able to understand their role in the target function [12].
- approximation of hypothesis maps in our brain into a trade-off between formulas that are precise but difficult to manage and approximate formulas to derive at least the first hints.
- accuracy of data maps into our attitude of "never shooting a sparrow with a cannon" or tracking a butterfly.

3 The MFE procedure

The core of the procedure is rather simple and quick to run. We may configure it as a function ϕ of three arguments:

1. the list ls of input variables to take into consideration,
2. the degree $polydeg$ of the polynomial through which to interpolate the marginals g_i of the target function g, and
3. the granulation rate gr, i.e., partition size according to which vector quantizing the input variables.

Namely,

– Each variable $x \in ls$ is sorted and partitioned in a group of size gr. Then each value in the group is replaced by the group mean. The outcome is $\tilde{x}$.
– A base learner h_i is computed as the regression between the target y and the selected $\tilde{x}_i$.
– The approximating h is the linear regressor of y again on the weak learners' outputs.

The use of h and the evaluation of its performance are carried out as usual. Rather some attention is deserved to the experimental environment to which the procedure is applied. The three strongholds are:

1. Neural network like an oracle. We assume having well trained a sufficiently powerful neural network to approximate the target function satisfactorily. Hence, like with an oracle, we may obtain a reliable answer on any input.
2. Explanation like the ones from domain experts. Since we are not interested in discovering the truth, rather in getting wisely understandable descriptions, we circumscribe the input domain to the field of interest.
3. Surgical bombardment of the interest field. The "bombs" are symbolic functions with which we fit a huge set of input/output pairs supplied by the neural network. According to their descriptional length, we drop them sequentially, which means in a progression with function degree and number of variables. Entropic criteria will decide the suitability of each bomb to be maintained or removed from the final arsenal.

4 Numerical experiments

We carried out two families of experiments. A former one is on artificial data to compare the efficiency of our method with a template of AIM style one [19]. A second one concerns real data available on UCI repository.

4.1 Artificial data

For comparison sake, we considered the simulation study S5 in [19], concerning the regression

$$y = x_1 - x_2 + \frac{2(x_3 + x_4 + x_5 + x_6)}{0.5 + (1.5 + x_3 + x_5 - x_4 - x_6)^2} + \epsilon \tag{2}$$

A (training-set,test-set) pair of size (10000,1000) has been generated, by starting with a uniform random seed in $[-1, 1]$ for each x_i to which a same bias constituted by a uniform random bias in $[-1, 1]$ was added to inject a correlation = 0.5 between each pair of variables. A final rescaling by a factor 0.5 was applied to gather the computed x_is in the range $[-1, 1]$, again. y_is are computed directly through (2). In Figure 2 we contrast the MFE architectures (on the left) with the one in the reference paper (on the right). The symbolic functions in the blue boxes are replaced by neural networks in the azure boxes. Moreover, while in the

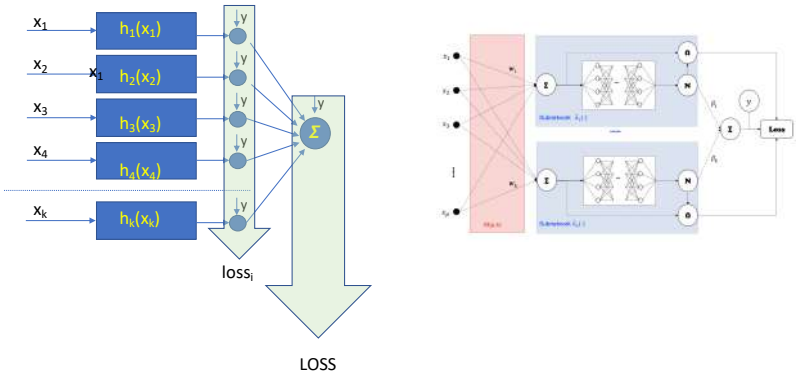

Fig. 2: The computational architectures. Left: MFE; $loss_i$ refers to the training of the base learners hi. $LOSS$ refers to the training of their linear combination Σ. Right: xNN architecture from [19]. w and β corresponds to β and γ in (1) respectively. $\mathbf{N}$ and $\mathbf{\Omega}$ are normalizing nodes and weighting nodes, respectively, $Loss$ refers to the overall training of xNN

former we have a double training, at weak learners' level and at their ensemble level, in the latter we have the usual training of a neural network, albeit with a distinguished structure of the network.

For training and test test size S $(10000, 10000)$ xNN claims a root mean square error on the test set $RMSE = 1.0049$ (actually a bit greater than the optimum of its competitors = 1.0005). Figure 3 displays our results as a function of the three parameters of the routine ϕ. A first thing we note is that $polydeg = 2$ surface almost coincides with the one of $polydeg = 3$ [7], both denoting RMSEs definitely lower than the xNN one. A second remark is on the apparently small sensitivity of the performance to the granulation.

Figure 4 left digs deeper into this aspect by showing an extensive course of this trend (gr from 1 to 1500). We note an almost insensitive RMSE growth for a granulation up to 200. In any case, even for a granulation 1500 MFE performance is better than the xNN one. Finally, Figure 4 right shows the scattering of the regressed values with respect to the original ones. The way of getting better accuracy will be dealt with in the next subsection in regard to natural data.

4.2 Real data

Collecting real data on complex phenomena such as the one mentioned in the Introduction is a costly task *per se*. Thus to move to the public UCI repository [6] and an application field that is the icon of *simplified* scenarios: Facebook transactions. Namely, the benchmark Facebook Comment Volume lists the number of comments in the first 24 hours after the publication of a post on Facebook as a

[7] Actually, in both cases we considered also a power $x^{1/2}$

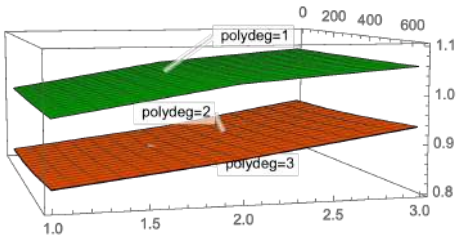

Fig. 3: MFE performance surfaces. z axis → RMSE, x axis → granulation size (ranging from 1 to 600, y axis→ left extreme of the indexes interval selecting the involved variables from the set $\{x_1, \ldots, x_6\}$. Surface label → degree of the regressed polynomials.

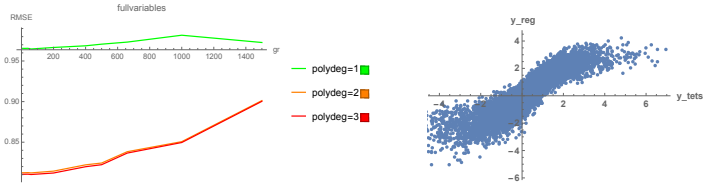

Fig. 4: Left: course of RMSE with the granulation size; parameters as in the labels. Right: scatter plot of the regressed versus original data of the test set, in the most favorable setting of the parameters (all variables used, $polydeg = 3$, no vector quantization).

function of 53 variables characterizing the post, such as "Page popularity" and "Page category." The number of records is $40,949$, the overall variables are 54, of which we considered only the first 33. This decision limits the problem's complexity (up to us in the end) at the cost of neglecting some ancillary variables. In greater detail, the data (extracted by a Web crawler) are:

- f1 number of likes for the source of the document.
- f2 number of individuals so far visited this page.
- f3 the daily number of activities by visitors to the page.
- f4 category of the source of the document.
- f5-f29 min, max, average, median, and standard deviation of features f30 to f33 plus the difference between f32 and f33.
- f30 The total number of comments before a pre-established date.
- f31 Number of comments in the last 24 hours before the pre-established date.
- f32 Number of comments in the last 48 to last 24 hours before the pre-established date.
- f33 Number of comments in the first 24 hours after the publication of the post, but before the pre-established date.

Finally, after deletion of less relevant variables based on p_values computed with the usual R regression tools [8], we come to the problem of computing f33 from f25 to f32. We tackled this problem in three modes

1. through a deep neural network
2. through MFE re the original f33
3. though MFE re the output of the deep neural network, where the latter plays the Oracle's role.

Moreover, we solved the problem at two sample scales: small (training set size =1000, testing set size =1000 → experiment size 2000) and large (training set size =10000, testing set size =10000 → experiment size 20000), overall working on the first 30.000 records of the dataset.

Deep neural network We trained an $8-30-32-1$ neural network (DNN) on the above dataset with standard Keras [9] options to baseline the learning difficulty of our task and build up the Oracle for mode 3. To this aim, we preferred limiting the input variables exactly to the 8 ones on which we will base the interpretation of the learned network. Given the asymmetry of the variable distributions, we escape any normalization and feature extraction (for instance, via PCA), but a uniform constraining of each variable's values in the interval $[-1, 1]$. Figure 5 reports the RMSE curves in recall as a function of the learning cycles and the original-reconstructed values scatter plots in both training and recall phases for the two sizings of training and test sets. Despite the regular course of RMSEs, with a noticeable improvement when we enlarge the training set, the first scatter plot, related to the smaller sample size, denotes overfitting with consequent lousy generalization on the test set. The second one is as expected. Hence we decide using the second neural network as an Oracle and register the value 0.028016 to be the RMSE of this Oracle.

[8] `https://cran.r-project.org`
[9] `https://keras.io`

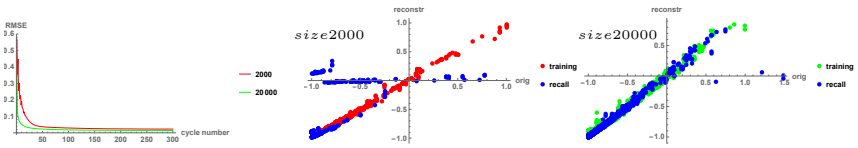

Fig. 5: Course of the loss descent and learning scatter plots of DNN. Parameters as in the labels.

Regressing the original data Our inference device is the same as the one used in the case study, with the sole exception of the number of input variables, now in number of eight. The first experiment with size 2000 shows that MFE doesn't face overfitting. The pictures in Figure 6, first row, denote a progressive degradation of the performance with the input granularity and the reduction of the number of input variables, but close trends of the error in training and recall. As for the degree of the regressing polynomials, we my see that in degraded conditions the linear regression may prove even more efficient than non-linear ones. We remark that the course of RMSE with the number of variables has an appreciable slope only when their reduction is over 3, thus allowing us to run our interpretation on 5 or less variables. The analogous of Figure 5 is in Figure 6, second row, where, since MFE is a one shoot procedure, in place of the error descent curves we show the RMSE surfaces as in the first row, now referred to an experiment size 20000.

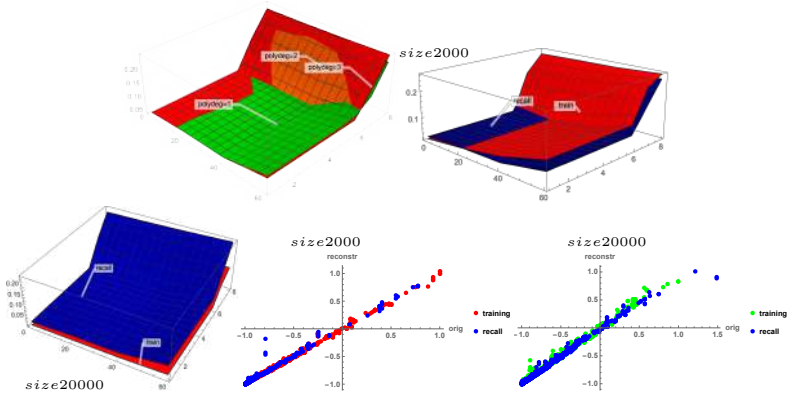

Fig. 6: First row: MFE performance surfaces for experiment size 2000. Left picture: same notations as in Figure 3; right picture: contrasting training and recall surfaces for polynomial degree = 3. Second row: MFE performance surfaces and scatter plots for different experiment sizes. Surface colors as in the first row; scatterplot colors as in Figure 5.

Regressing the Oracle data If we repeat the experiment in the previous section apart for the target, now replaced by the output of the DNN trained on 10000 examples (in the role of the Oracle), we obtain error surfaces that are very close to those in Figure 6. Figure 7, left side, contrasts the recall RMSE surfaces of the two experiments. To summarize the course of RMSE, in Table 1 we report these values in training and recall for DNN and for MFE runs in similar conditions, i.e. using all eight variables and with no approximation on their values (let us call them *ideal conditions*). We note that MFE outperforms NN not only when running on the same sample size but also, as for recall, on the smaller sample size. Inferring MFE on the NN output may suffer from a slight overfitting that disappears when we abandon ideal conditions, as shown in Figure 7, left side. However, this overfitting is very limited, as shown in the right picture of the figure.

RMSE	NN10000	MFE1000	MFE10000	MFEOrac
training	0.014272	0.0187171	0.0108914	0.0094463
recall	0.028016	0.0274221	0.0137267	0.0211474

Table 1: Contrasting the performances of the methods considered in the paper

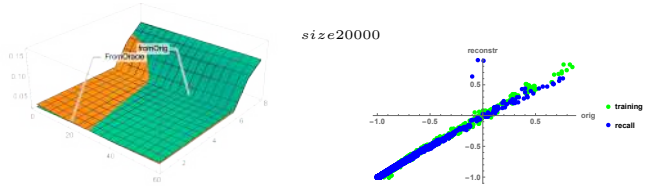

Fig. 7: Left: contrasting the RMSE surfaces computed on the original test set and on the DNN generated one; right: reconstruction scatterplots of the second target in *ideal conditions*. Notations as in Figure 6.

5 Discussion and concluding remarks

Expressing a complex function by means of a combination of ridge functions is a recurrent modeling pattern across scientific domains. We started from the notion that its success as a modeling device may be due to the fact that it provides a dimensional decomposition of complex functions that naturally facilitates human understanding and interpretation of the underlying phenomenon. These functions can be seen as modular components of the expression of the overall

phenomenon; the fact that they are symbolic and can be learned independently and at different times makes our approach a potential (interpretable) alternative to multi-stage learning for achieving model adaptation. In a non-stationary situation, we could retrain our model piece-wise, using different retraining windows to handle the selective obsolescence of knowledge represented by the individual atomic function or new domain knowledge typical of transfer learning [21]. Being each atomic component symbolic, its phase-in/phase-out lifecycle would be understandable for humans. We plan to investigate using the performance gap between different base learners' configurations h_is as a measure of the discrepancy between the source and target domains [17].

According to [1], "there is no standard and generally accepted definition of explainable AI. The XAI term tends to refer to the many initiatives and efforts made in response to concerns regarding AI transparency and trust concerns, more than to a formal technical concept". These initiatives aim to *white-boxing* AI, i.e., gaining a quantitative understanding by humans of AI models' operation [10], where deep neural networks represent the paradigmatic target. Unfortunately, deep learners generally prove as powerful to learn hard functions as impenetrable to any form of understanding. With similar targets, we assume quantitative understanding to be an unavoidable premise to achieving human-understandable explanations (for instance, in simplified natural language). We argue that quantitative interpretations' accuracy measures are not a guaranty but a concrete prerequisite of a satisfactory approximation of any conceptual explanation. This assumption contrasts the popular (but in our opinion debatable) idea that an explanation that is not accessible to laypeople is defective by definition. As for a climbing trip to the Alps, to reach the peak, we must be trained.

Our experimental results support the idea that our *symbolically driven* approach may provide the level of accuracy suitable for practical applications as a complementary interpretable alternative to deep learning architectures.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (xai). IEEE Access **6**, 52138–52160 (2018)
2. Apolloni, B., Chiaravalli, S.: PAC learning of concept classes through the boundaries of their items. Theoretical Computer Science **172**, 91–120 (1997)
3. Apolloni, B., Kurfess, F. (eds.): From Synapses to Rules – Discovering Symbolic Rules from Neural Processed Data. Kluwer Academic/Plenum Publishers, New York (2002)
4. Apolloni, B.,Shehhi, A., Damiani, E.: The simplification conspiracy. In: Progresses in Artificial Intelligence and Neural Systems. Smart Innovation, Systems and Technologies. pp. 11–23. Springer-Verlag, Lecture Notes in Artificial Intel (2019)
5. ApolloniB., Bassis, S., Malchiodi, D., Pedrycz, W.: The Puzzle of Granular Computing, Springer, 2008, in press., vol. 138. Springer-Verlag, BERLIN – DEU (2008)

[10] `https://www.iarai.ac.at/research/a-theory-of-ai/white-boxing-interpretability/`

6. Corke, P.I.: A robotics toolbox for matlab. IEEE Robotics Automation Magazine **3**(1), 24–32 (1996)
7. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems **2**(4), 303–314 (1989)
8. Garey, M.R., Johnson, D.S.: Computer and Intractability: a Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco (1978)
9. Hastie, T.J., Tibshirani, RJ: Generalized additive models. London: Chapman & Hall (1990)
10. Josephson, J.R., Josephson, S.G.: Abductive Inference: Computation, Philosophy, Technology. Cambridge University Press (1994)
11. Kelley, T.D.: Symbolic and sub-symbolic representations in computational models of human cognition: What can be learned from biology? Theory & Psychology **13**(6), 847–860 (2003)
12. Kohlhase, A., Kohlhase M., and Fuersich, M. :Visual structure in mathematical expressions. Proceedings of the International Conference on Intelligent Computer Mathematics. Springer, Cham, 2017.
13. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. ArXiv **abs/1706.07269** (2019)
14. Ruan, L., Yuan, M.: Dimension reduction and parameter estimation for additive index models. Statistics and Its Interface **4** (01 2010)
15. Vapnik, V.: Estimating of dependencies based on empirical data. Springer, New York (1982)
16. Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J.J., Nair, V.: Explainable neural networks based on additive index models. ArXiv **abs/1806.01933** (2018)
17. Wang, B., Mendez, J., Cai, M., Eaton, E.: Transfer learning via minimizing the performance gap between domains. In: Advances in Neural Information Processing Systems, vol. 32, pp. 10645–10655 (2019)
18. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
19. Yang, Z., Zhang, A., Sudjianto, A.: Https://arxiv.org/pdf/1901.03838.pdf
20. Yuan, M.: On the identifiability of additive index models. Statistica Sinica **21**, 1901–1911 (2011)
21. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE pp. 1–34 (2020)