Scientific Machine Learning: Towards Predictive Closed-Form Models

Dimitri Papadimitriou and Steven Latre

University of Antwerpen, Antwerpen, Belgium,
dimitri.papadimitriou@uantwerpen.be

Abstract. Captured from ecological or natural systems, high-dimensional scientific data span a large spectrum ranging from physical to bio-chemical processes. Though machine learning plays nowadays a role in automating various related data analysis and mining tasks such as pattern recognition, feature extraction (detection, identification, etc.) and dimensionality reduction tasks, data alone are not sufficient to understand phenomena. If the aim is actually to understand underlying mechanisms and processes, then models are needed. The question becomes thus which models to build and how to build these models ? This paper addresses these questions and related challenges from the perspective of neural-based learning principles and techniques.

Keywords: neural networks, domain uninformed problems, partial differential equations

1 Introduction

Scientific disciplines mainly focus on exact mathematical models obtained by combining first principles (natural laws, interactions, etc.) with experimental data although approximate models (of lower complexity) could be obtained from measurement data that allow to tune complexity vs. accuracy. Measurement data being imprecise due to measurement errors, often noisy and possibly generated by complex phenomena, are not exactly representable by a model in the considered hypothesis class. Thus, both statistical estimation and deterministic approximation aspects should be present in data modeling related tasks though often dominated by the former (as further detailed in Section 2) [17].

The involvement of machine learning in explanatory but also predictive modeling remains mainly driven by pre-existing physical model knowledge: hypothesis are drawn from function spaces/model classes derived from and delimited by knowledge acquired by external means. Indeed, application of machine learning to scientific disciplines, focuses so far primarily on forward problems (predict the response u of the system from model parameters m) and inverse problems (identify/estimate the model parameters m from observations of its response u). Both problems are often formulated by assuming that the underlying phenomenon Φ is a dynamical (data generating) system characterized by sets of partial differential

equations (PDEs), that are typically non-linear and non-homogeneous. In both cases the (non)linear differential operator is assumed to be known but the PDEs parameters (coefficients, source terms, etc.) have to be identified. However, these parameters are often hardly or even not at all accessible to direct measurements: they have to be fitted from indirect measurements. In turn, the involvement of machine learning focuses on solving inverse problems of parameter identification [25] that are mathematically challenging since often not well-posed in the sense of Hadamard [11] [1].

Hypothesis generation and model identification, i.e., automatically finding the explicit formulation of the (nonlinear) operator $\mathcal{F}$ mapping the model parameters m to the system's response/solution $u = \mathcal{F}(m)$, stays outside the main application scope of machine learning. This is the case when the dynamics of the underlying phenomenon Φ is assumed to be characterized by sets of PDEs; especially when their explicit formulation is unknown, i.e., the available data doesn't follow a given or known physical model. Indeed, identifying from high dimensional spatio-temporal data the mathematical expression of the nonlinear PDEs that governs the dynamics/evolution of the data is even less attainable with current machine learning methods and techniques. As we will show throughout this paper, operating in closed and adaptive loop to learn the physical laws, models and structures underlying the data remains to be accomplished. Hence, for scientific disciplines, machine learning is (still) not used to build a mathematical model of the underlying phenomenon but mainly as a pre-existing model fitting or statistical inference tool.

On the other hand, machine learning and neural learning techniques in particular don't look so far at scientific disciplines with a specific lens. Compared to computer vision for instance, where neural learning has both adapted its models (e.g., convolutional neural networks, Cayley neural networks) and specialized training algorithms (e.g., pooling, iterative total variation methods), nothing similar can be recorded for scientific disciplines. To bring machine learning at the level of a mathematical modeling tool for scientific disciplines, referred to as Scientific Machine Learning (SML), a broader set of techniques ranging from mathematical model building until their validation and verification shall be involved.

Few trials to fill this significant gap have been recently initiated, e.g., [7]. They often start by considering a broad set of statistical learning challenges and hope that their potential solving could find some traction or applicability. However, SML is not about overcoming the well-known fundamental limits of statistical learning or generic machine learning techniques but about understanding and solving scientific problems by involving specific/customized machine learning techniques when generic ones fail to address them or are simply unavailable (thus, requiring new techniques).

The remainder of this paper is structured as follows. Section 2 positions and details the proposed modeling method together with the possible role and cur-

[1] Note also that parameter identification problems are often nonlinear even if the PDE itself is linear

rently identified limits of statistical machine learning. Related and prior research work are documented in Section 3 and contrasted against already identified fundamental tradeoffs. Section 4 documents the essential techniques involved in the scientific machine learning context. Finally, Section 5 draws some concluding remarks and perspectives for this nascent research domain.

2 Predictive Closed-Form Modeling

For bringing machine learning as a mathematical modeling tool for scientific disciplines, its primary goal shall move beyond solving inverse problems and focus on discovering hidden or even new physical models and laws underlying the dynamics of, e.g., natural/ecological systems and its various bio-physical processes. The overall method consists of automatically transforming the high-dimensional spatio-temporal data obtained from observation into predictive mathematical models without assuming that the available data follows a given or known physical law/model described by a system of partial differential equations.

2.1 Positioning

As outlined in the Introduction, this problem –and related challenges– contrasts to the current application scope of statistical machine learning and related methods in the context of physical and more generally natural sciences. This scope is still mainly focused on the solving of domain-informed/explicit inverse problems by means of data modeling methods [5]. They rely on the assumption that the data (representing the response u of the system to independent input variables x) are generated by a given stochastic model. These methods enable to formulate hypothesis and estimate the model that best explains the data (fitting pre-existing physics-based models to the data) but also perform various inference tasks (such as density estimation) from data samples (x, u).

As data obtained from observations of dynamical systems involving unknown physical, bio-chemical or ecological mechanisms show increasing complexity, data models specification would also become more complex up to losing the advantage of providing a (relatively) simple explanation of the underlying natural phenomena (if ever explainable by parametric models). Hence, in addition to statistical/quantitative methods, solving such problems may also involve –but to a lesser extent– qualitative methods. Involving the latter enables to approximate high-dimensional process by a small number of free parameters by extracting local and global intrinsic features and structures as well as their relations that are invariant under various transformations or embeddings.

Heavily influenced by data models due to their explanatory power (with the assumption that models with high explanatory power possess inherently high predictive power), physical/natural sciences have not widely exploited algorithmic models so far. Algorithmic modeling methods include among others deep neural networks. They aim at finding a transformation function of the input x, i.e., an algorithm that operates on the input data to produce/predict the

response u while treating the data mechanism as unknown. Though often lacking interpretability, algorithmic methods are a priori suitable for the automated learning of predictive models capable to forecast with high accuracy the evolution of physical systems.

Nevertheless, discovering the mathematical expression of predictive models even with algorithmic methods remains highly challenging. Especially, when the high-dimensional spatio-temporal data that is available doesn't follow a pre-existing/known physical law or model. In this case, searching for an (approximate) model of the physical process Φ involves the generation of hypothesis from a function space that isn't necessarily limited anymore by these pre-existing/known physical models. Hence, some constraints have to be considered to guide the search process. For instance, fixing the highest derivatives order (instead of considering it as a free parameter) would limit the dimensionality of the search space.

Compared to classical physics- or more general domain-informed inverse problems, such class of problems can be referred to as domain-uninformed [2]. Although inverse problems span nowadays implicit variants when data cannot be formulated explicitly as a function of the unknown model parameters, probabilistic methods through Monte Carlo sampling of feasible solutions of implicit inverse problems or local search matheuristics yield numerical procedures.

2.2 PD(A)E Modeling

To uncover/extract the physical laws from high dimensional spatio-temporal data obtained from potentially noisy observations, the learning task consists of identifying the mathematical expression of the nonlinear partial differential equations (PDE) that governs the dynamics/evolution of the observed data. Nonlinear dynamics implies that the system is not assumed a priori to satisfy the superposition principle. For this purpose, we assume throughout this paper that the underlying physical system is governed by coupled equations of the form:

$$\begin{cases} u_t &= F(x, y, u, v, u_x, u_y, v_x, v_y, u_{xx}, u_{yy}, v_{xx}, v_{yy}, \ldots) \\ v_t &= G(x, y, u, v, u_x, u_y, v_x, v_y, u_{xx}, u_{yy}, v_{xx}, v_{yy}, \ldots) \end{cases} \tag{1}$$

In this system, the nonlinear functionals F and G depend on

- the solutions $u = u(x, y, t)$ and $v = v(x, y, t)$ function of time t and multi-dimensional vectors $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_m) \in \Omega \subset \mathbb{R}^m$
- their first-order time derivatives $u_t = \frac{\partial u}{\partial t}$ and v_t defined similarly
- their element-wise first-order partial derivatives $u_x = \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_m}$, u_y, v_x, and v_y defined similarly

[2] This class of problems could also be referred to as model- or physics-free though this terminology is avoided for trivial reasons

- their element-wise second-order partial derivatives $u_{xx} = \frac{\partial^2 u}{\partial x_1^2}, \ldots, \frac{\partial^2 u}{\partial x_i \partial x_j}, \ldots, \frac{\partial^2 u}{\partial x_m^2}$, u_{yy}, v_{xx}, and v_{yy} defined similarly
- etc.

Note that the solutions u and/or v can also be multi-dimensional, i.e., $u = (u_1, \ldots, u_p)$ and $v = (v_1, \ldots, v_q)$. In this case, u_x denotes the component-wise elements of the Jacobian $\frac{\partial(u_1,\ldots,u_p)}{\partial(x_1,\ldots,x_m)}$ and u_{xx} the components of the divergence of the Jacobian (a.k.a. vector Laplacian), and similarly for v_x and v_{xx}. This system of equations could be further generalized by involving, e.g., external force field, bifurcation parameters. As in general the system of PDEs is often incomplete, including a set of algebraic equations closes the system to form a partial differential algebraic equation (PDAE) set. Hence, throughout this paper, such models are referred to as closed-form.

This system of equations is quite general and can be used for modeling many physical processes in various domains and environments including ecology, climatology, geophysics, hydrology, fluid dynamics, etc. but also neuro-physiology. Observe that the Fitzhugh-Nagumo neuron model (simplified version of the Hodgkin-Huxley model) is governed by a system of equations of this form.

2.3 Role of Machine Learning

Common statistical machine learning techniques applied to finite data samples find limited applicability in the context of predictive modeling. The main reason being that physical processes often verify at least one if not all of the following properties that violate their key working assumptions:

1) *non-linear*, sometimes even the type of nonlinearity is unknown, i.e., their dynamics cannot be modeled linearly due to, e.g., phase transitions, saturation effects, asymmetric cycles, chaotic behavior, etc.; although some (covariance-stationary) processes admit a Wold representation (sum of a deterministic component and an infinite sum of noise terms) that looks linear;
2) *non-stationary* their statistical properties are time-dependent including but not limited to, e.g., non-constant mean, variance as well as possible trends, periodicity or seasonality (sequential data over long time periods);
3) *non-mixing*, i.e., the statistical dependence between events does not reach 0 as time increases.

Neural networks, thanks to their nonlinear function approximation capacity have the potential to address -at least partially- the former. When handling temporal data that show different variation at different levels, their transformation (e.g., Box-Cox, Fisher transform) can help stabilizing the variance. First- and second-order differences at lag 1 and seasonal differences can help stabilizing the mean; thus, reducing trend and seasonality. These operations are commonly applied in the analysis and modeling of time series, particular case of difference equations. In comparison, fitting continuous time models (i.e., time-dependent PDEs) to the data observed from non-stationary non-mixing processes remains

challenging, especially when distributional assumptions can't be formulated. The predictive capacity of the model renders this task even more difficult. For instance, with neural networks the target model would be approximated globally since training data (observations) are generalized during the learning phase. Instead, one might consider a learning method that produces different local approximations in the target model and generalization beyond observation data delayed until a prediction request is made.

On the other hand, empirical data samples do not necessarily verify the i.i.d. assumption at the root of statistical learning techniques, in particular, when training/calibration dataset and testing/running dataset do not follow the same distribution. Related measurement (variables) are affected by random errors that violate key assumptions of OLS fitting since often non-Gaussian and heteroskedastic (their variance is not constant instead of being unrelated to any of the explanatory/independent variables). Last but not least, accuracy (closeness of agreement between measured value and true value -when known) and precision level at which predictions have to be realized play an essential role in the selection/investigation of the appropriate (class of) technique(s).

Consequently, although the availability of often large sets of labeled data pairs $\{(x,t;m),u\}$ would imply the straightforward execution of existing model selection and identification techniques to find $\mathcal{F}$ such that $u=\mathcal{F}(x,t;m)$, their criteria of applicability may not be verified for the classes of problems under consideration. Although neural networks can be trained to obtain the numeric approximation of the relationship between the independent variables x and the various parameters m of the underlying process, determining the mathematical expression that would translate such relationships remains to be addressed.

2.4 Example

In this section, we present an example of how physics-based and measurement-based modeling would model fluxes of greenhouse gas (GHG) that play an essential role in climate change [13]. In atmospheric transport, air motion can be modeled as an horizontal flow of numerous rotating swirls (called Eddies) of various sizes. Each Eddy has 3-D components including vertical air movement/vertical wind. The modeling of molecular and turbulent mass transport starts from the advection-diffusion equation describing how the concentration $c=c(x,y,z,t)$ varies with time:

$$\frac{\partial c}{\partial t}+u\frac{\partial c}{\partial x}+v\frac{\partial c}{\partial y}+w\frac{\partial c}{\partial z}=\frac{\partial c}{\partial x}(D_x\frac{\partial c}{\partial x})+\frac{\partial c}{\partial y}(D_y\frac{\partial c}{\partial y})+\frac{\partial c}{\partial z}(D_z\frac{\partial c}{\partial z}) \qquad (2)$$

Following the Reynolds decomposition (averaging rule), the instantaneous velocity u_i along axis i can be decomposed as $u_i=\langle u_i\rangle+u_i'$ (a) where $\langle u_i\rangle$ is the time averaged velocity and u_i' the instantaneous fluctuation component (deviation from mean value) and the instantaneous concentration as $c=\langle c\rangle+c'$ (b) where $\langle c\rangle$ is time averaged concentration and c' the instantaneous fluctuation component (deviation from mean value), with $\langle c'\rangle=0$.

Substituting (a) and (b) in Eq.2 with incompressible flow assumption $\frac{\partial\langle u_i\rangle}{\partial x_i} = 0$, homogeneous diffusion coefficients, yields the time averaged equation:

$$\frac{\partial\langle c\rangle}{\partial t} + u_i\frac{\partial\langle c\rangle}{\partial x_i} = D\frac{\partial^2\langle c\rangle}{\partial x_i^2} - \frac{\partial\langle u_i'c'\rangle}{\partial x_i} = \frac{\partial}{\partial x_i}\Big(D\frac{\partial\langle c\rangle}{\partial x_i} - \langle u_i'c'\rangle\Big) \tag{3}$$

The second term in the RHS of Eq.3 corresponds to the turbulent flux arising from the correlation between u_i' and c' : $\langle u_i'c'\rangle = \langle u_ic\rangle$–$\langle u_i\rangle\langle c\rangle$. Since the coefficient D is usually a very small quantity, the molecular transport term can be neglected compared to the turbulent transport. Hence, we obtain:

$$\frac{\partial\langle c\rangle}{\partial t} + \langle u\rangle\frac{\partial\langle c\rangle}{\partial x} + \langle v\rangle\frac{\partial\langle c\rangle}{\partial y} + \langle w\rangle\frac{\partial\langle c\rangle}{\partial z} = -\frac{\partial\langle u'c'\rangle}{\partial x} - \frac{\partial\langle v'c'\rangle}{\partial y} - \frac{\partial\langle w'c'\rangle}{\partial z} \tag{4}$$

This PDE cannot be solved exactly because the time averaged equation comprises 3 new unknown quantities (leading to more unknowns than number of equations). Three methods can be considered to overcome this problem:

- **Physics model method**: approximates the turbulent fluxes $\langle u'c'\rangle$, $\langle v'c'\rangle$ and $\langle w'c'\rangle$ by their mean concentration gradient (following the Ficks law):

$$\langle u'c'\rangle = -\epsilon_x\frac{\partial\langle c\rangle}{\partial x}, \langle v'c'\rangle = -\epsilon_y\frac{\partial\langle c\rangle}{\partial y}, \langle w'c'\rangle = -\epsilon_z\frac{\partial\langle c\rangle}{\partial z} \tag{5}$$

  The coefficients ϵ_x, ϵ_y and ϵ_z (assumed $\gg D$) are determined by, e.g., numerical simulation and the modeling equation retrieves the form of a canonical advection-diffusion equation (with c denoting $\langle c\rangle$ and u_i denoting $\langle u_i\rangle$):

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} + v\frac{\partial c}{\partial y} + w\frac{\partial c}{\partial z} = \frac{\partial}{\partial x}\Big(\epsilon_x\frac{\partial c}{\partial x}\Big) + \frac{\partial}{\partial y}\Big(\epsilon_y\frac{\partial c}{\partial y}\Big) + \frac{\partial}{\partial z}\Big(\epsilon_z\frac{\partial c}{\partial z}\Big) \tag{6}$$

- **Measurement-based method**: estimates the turbulent fluxes from measurement. From the observation of the measurement variables w' and c' compute the vertical flux F_z along the $z-$axis as the mean covariance product $\langle w'c'\rangle$ between the deviation (from time averaged value) of the instantaneous vertical wind speed w' and the deviation (from time averaged value) of the instantaneous concentration c'. Thus, model unknowns appear as (composition of) measurement variables.
- **Machine learning-based method**: approximates the solution by means of a multi-layer neural network (trained with a pre-defined range of coefficient values obtained, e.g., from simulation). Then, use the (numeric) solution $\hat{c}$ as input for the solving of an inverse problem to determine the diffusivity coefficients ϵ, i.e., $\hat{c} = \mathcal{F}(\epsilon)$. Since the operator is nonlinear, solving the inverse problem relies on an iterative method based, e.g., on the Levenberg-Marquardt algorithm. This combined forward-inverse loop may also be repeated iteratively or tuned until obtaining a solution satisfying quality criteria. This method extends to the case where the PDE model itself is unknown and only its numerical approximation by the neural network is of interest. However, extracting the mathematical expression of the PDE model itself remains to be tackled as explained in Sections 3 and 4.

3 Related Work: Main Approaches and Tradeoffs

The first method proposed by [23] to uncover the mathematical expression of the PDE model consists of constructing a dictionary containing a large collection of candidate atomic terms (partial derivatives) that are likely to appear in the unknown functions F and G. Then, by taking advantage of sparsity-promoting techniques such as sparse nonlinear regression, one can determine the coefficients of the terms appearing in the expansion of the unknown functions F and G. Thus, determine the most informative/significantly contributing terms in the RHS of the partial differential equations that most accurately represent the time dynamics of the data. This method leverages the fact that most physical systems have only a few relevant terms that define the dynamics, making the governing equations sparse in a high-dimensional nonlinear function space. However, dictionary learning is typically limited to small-scale representation problems whereas the required number of terms to include a priori (i.e., at training time) in the library increases exponentially with the dimensionality of the input data, the output/solution as well as the derivatives order. Observe also that in comparison symbolic differentiation suffers from expression swell as it can easily produce exponentially large symbolic expressions which take correspondingly long to evaluate. Hence, the main challenge becomes to avoid the computational limits of numerical differentiation while preserving interpretability of the learned dynamics (that is unknown beforehand) through its mathematical expression. Changing the dictionary such that the system of equations is expressed in a single multi-vector equation using geometric algebra could be a viable alternative at the condition that the resulting model remains interpretable.

The main alternative exploits the nonlinear function approximation property of (feedfoward) multi-layer neural networks. The straightforward method as proposed in [22] would proceed by representing the unknown solutions u and v as well as the nonlinear functions F and G each by a multi-layer neural network. However, exploiting the expressivity of neural networks comes at the cost of completely losing the interpretability of the learned functions F and G defined by Eq.1, since their functional form remains unrevealed. Thus, this approach keeps the physical laws underlying the dynamics of the data generating system/physical process hidden. Hence, such method introduces a fundamental tradeoff between expressivity and interpretability in addition to the usual tradeoff between expressivity and trainability of neural networks (i.e., learnability of their parameters). Obviously, one could fall back onto physics-informed methods to inverse problems but then one should relax model assumptions, otherwise uncovering new physical laws from observation data would remain unaddressed.

Next to direct methods, an indirect method has been recently proposed in [18] that develops a lifting technique for nonlinear system identification/parameter estimation based on the framework of the Koopman operator theory [15]. In general, indirect methods solve an initial value problem and do not require the estimation of time derivatives [4]; they provide an alternative to direct methods at the expense of solving a (nonconvex) nonlinear least squares problem. Instead of identifying the nonlinear system in the state space, the key idea developed is

to identify the linear infinite-dimensional (linear) Koopman operator in the lifted space of observables, a process which results in a linear method for nonlinear systems identification. Hence, this indirect method not only circumvents the estimation of time derivatives but also relies on linear least squares optimization. It provides a parametric identification technique that can accurately reconstruct linear/polynomial vector field of a broad class of systems (including unstable, chaotic, and system with inputs). Its dual provides estimates of the vector field at the data points and is well-suited to identify high-dimensional systems with small datasets. Extension of this lifting method for identifying nonlinear PDEs though potentially attractive for low-sampling rate datasets remain centered on identification / nonlinear parameter estimation; hence, subject to the same limits experienced by direct parametric methods using library functions in addition to its inability in identifying general vector fields.

Consequently, a workable and provable technique to uncover the closed-form expression of predictive PD(A)E-based models remains clearly beyond reach of existing machine learning methods including neural networks.

4 Neural Network-based PDE Learning

New insights are required to bring machine learning at the level of a data-driven mathematical modeling tool capable to automatically discover the closed-form expression of the PD(A)Es that govern the dynamics of various natural/ecological systems (as observed from the data). For this purpose, this section focuses on how the main learning task could be realized by means of advances in neural networks for the reason explained in Section 2.3.

4.1 Neural Networks as PDE Solvers

Though approximating solution of nonlinear PDEs by (deep-)neural networks is not the final objective when learning the closed-form expression of PD(A)E models, such class of solvers constitute an essential part of the proposed modeling method. The theoretical foundation for approximating a function and higher-order derivatives with a neural network relies on a variant of the universal approximation theorem by Hornik [12]. Theorem 4 in [12] establishes that if the activation function is $k-$times continuously differentiable, bounded and non-constant, then any $k-$times continuously differentiable function and its derivatives up to order k can be uniformly approximated by two-hidden layer neural networks on compact subsets. Even if rather restrictive, most recent papers show the existence of neural networks approximating solutions of the PDEs by (sometimes implicitly) referring to this Theorem.

In [24], authors propose a learning algorithm, referred to as Deep Galerkin Method (DGM), for approximating solutions of high-dimensional quasi-linear parabolic PDE. The DGM algorithm is similar in spirit to Galerkin methods, with the solution approximated by a neural network instead of a linear combination of basis functions. To satisfy the differential operator, initial conditions,

and boundary conditions, the training algorithm performs, instead of forming a mesh, on sequences of randomly sampled space and time points. This technique yields a meshfree method, which is essential for high-dimensional PDEs.

Despite these promising results and their potential extension to hyperbolic, elliptic, and partial-integral differential equations, several open research questions remain to be addressed:

- PDEs with highly non-monotonic or oscillatory solutions may be more challenging to solve and further developments in terms of neural network architecture are necessary. Indeed, neural model selection, including its depth, layer width and activation function requires to account for nonlinear functions of the input that rapidly change in certain time and space regions. Moreover, only under certain growth and smoothness assumptions on the nonlinear terms, the convergence proof follows by the smoothness of the neural networks together with compactness arguments.
- Solving the corresponding initial and boundary value problem (iBVP) relies on the penalty method which converts the constrained error minimization to an unconstrained problem using a first-order iterative method. The main drawback being that this method yields ill-conditioned problems when naively selecting penalty coefficients. The training algorithm appears thus as a main bottleneck for the solving of nonlinear PDEs. Instead, one should consider iteratively solving this error minimization problem using the augmented Lagrangian method. The latter adds a term corresponding to an estimate of the Lagrange multiplier in the objective of the unconstrained problem. Although, this method avoids ill-conditioning as the accuracy of the Lagrange multiplier estimation improves at every step, it not yet established if this algorithm will scale to more three-dimensional problems (domain $\Omega \in \mathbb{R}^3$).
- Neural network trained following this algorithm converge in L^p ($p \leq 2$) to the solution of the PDE as the number of hidden neuron units tends to infinity; hence, deep neural networks do not circumvent/avoid the curse of dimensionality problem though they can avoid forming a mesh. Moreover, upper bounds for the number of weights/neuron units and layers increase exponentially with the dimension d of the input space. Such bounds have been obtained when approximating, e.g., Sobolev-regular functions with respect to (fractional) Sobolev norms and Besov-regular functions, by deep neural networks; even if for these classes of functions deep neural models can achieve an optimal rate of approximation error.

On the positive side, the solving of nonlinear PDEs using neural networks with rectified linear units (ReLU) has been recently proven to achieve for both univariate and multivariate real-valued functions[3], the approximation error with respect to Sobolev norms (compared to the usual L^p norms, $p \in [1, \infty]$) and the approximation rate bounds fidelity achievable with hp-FEM. The latter is a

[3] Note: at the time of writing only the demonstration for the univariate case has been published in [21]

general version of the finite element method (FEM) developed by Babuska in the early 90's [2]. In this seminal paper, Babuska et al. discovered that FEM converges exponentially fast when the mesh is refined using a suitable combination of h-refinements (dividing elements into smaller ones) and p-refinements (increasing their polynomial degree). Although, from the approximation theoretical perspective, deep neural networks perform as well as the best available FEM method, the (uninformed) modeling/identification of nonlinear PDEs by means of neural networks still remains highly challenging.

4.2 Polynomial Neural Networks

For recognition/identification tasks, neural networks identify patterns according to their relationship, responding to related patterns with a similar output by transforming the weighted sum of inputs to describe overall similarity relationships of input patterns. Their main shortcoming in pattern recognition/identification is the disability of input pattern generalization: in case of rotation or translation of the input matrix of variables, the recognition/identification will fail. Polynomial Neural Network (PNN) [16] [27] for identification of dependence of variables describe a functional dependence of input variables –not entire patterns. A neural network can thus be seen as a simplified form of PNN [27] for which combinations of variables are missing. Polynomial functions apply these combinations to preserve partial dependencies of variables compared to conventional neural networks which can't utilize data relations for recognition/identification tasks.

Further, sum fraction terms can approximate multi-variable functions on the basis of discrete observations enables replacing a general PDE definition with polynomial elementary data relation descriptions. Differential polynomial neural networks (D-PNN) introduced by Zvajka in 2011 [28] form a class of neural networks, which construct and solve an unknown general PDE of a function of interest with selected substitution relative terms using non-linear multi-variable composite polynomials. The layers of the network generate simple and composite relative substitution terms whose convergent series combinations can describe partial dependent derivative changes of the input variables.

The starting point of the D-PNN development relies on the Group Method of Data Handling (GMDH) developed by Ivakhnenko [14].

– **Group Method of Data Handling (GMDH)**: iterative method introduced by [14] which decomposes the complexity of a system or process into many simpler relationships each described by a processing function of a single neuron. This method identifies general non-linear relations between input and output variables by means of support functions. Most GMDH algorithms use polynomial support functions expressed in the form of a functional Volterra series whose discrete analogue is known as the Kolmogorov-Gabor (K-G) polynomial

$$y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk} x_i x_j x_k + \dots \qquad (7)$$

where $(x_1, x_2, \dots, x_n)$ denotes the vector of input variables and $(a_1, a_2, \dots, a_n)$ the vector of coefficients. Each relationship is described by a low order two-variable polynomial processing function of a single neuron node.

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \qquad (8)$$

- **Polynomial Neural Network (PNN)**: enable the identification of variables dependence by describing a functional dependence of input variables (not entire patterns as DNN does). This could be regarded as a pattern abstraction in which identification is not based on values of variables but only on their relations. Multi-layer neural networks (including deep neural networks) can thus regarded as a simplified form of PNN where combinations of variables are missing and thus can't utilize data relations for recognition/identification tasks. Instead, polynomial functions apply those to preserve partial dependencies of variables. The GMDH algorithm builds up a polynomial (actually a multinomial) combination of the input components by forming in successive steps a PNN: adding one layer at a time, calculating its polynomial coefficients and selecting the best-output polynomial neurons, entered the next layer, so far as possible improvements in the last added output layer. The PNN structure is developed through learning: the number of layers of the PNN is not fixed in advance but becomes dynamically meaning as this self-organizing network grows over the trained period [19].
- **Differential PNN (D-PNN)** uses a complete multi-layer PNN structure to form (decompose) and solve an unknown general PDE of a searched function $u = f(x_1, \dots, x_n)$ with selected combination of substitution terms using non-linear/factional multi-variable composite polynomials. The layers of the network generate composite substitution terms whose convergent series combinations can describe partial dependent derivative changes of the input variables. The substitution relative derivative terms are selected and combined according to the similarity dimensional analysis, which forms system characteristics from dimensionless ratio groups of variables [20].
  Although, D-PNN extends the complete GMDH network structure to produce convergent sum series of relative derivative terms, which can define and substitute for an unknown general PDE of a searched multi-variable function model, the D-PNN's operating and design principles are different from those of GMDH. D-PNN decomposes the general PDE analogously to GMDH performs the general input–output connection polynomial (7). It applies the complete PNN skeleton to generate convergent sum series of selected substitution derivative terms using the principles of similarity analysis to solve the general PDE. Similarity analysis facilitates substitutions for differential equations or can form dimensional units from data samples to describe real-world problems. K-G polynomials (8) are suitable for modeling polynomials

with PNN. Instead, for a polynomial approximation of complicated multivariable periodic functions or time-series functions, the PNN approximation ability has to be improved by modifying some GMDH polynomial (7) items with the use of the sigmoidal function. The sigmoidal function may transform some polynomial items together with the parameters to improve the polynomial derivative term series ability to approximate complicated periodic functions. The simple reason stems because low order polynomials are not able to fully make up for the complete cycles.

Though at first glance attractive, the main drawback of D-PNN in modeling complexity is their direct proportionality to the increasing number of input variables, as further hidden layers are required to define all the possible combination PDE terms. Compared to conventional neural networks, Though such structures can form, more complex and versatile model types of systems described by a large number of data variables, the D-PNN complexity becomes overwhelming due to combinatorial nature of the compositions. Hence, progressing such techniques involves the progressive/iterative construction of the search space, thus the dynamic construction of the neural network itself.

4.3 General-purpose Automatic Differentiation (GAD)

Classical algorithmic data modeling methods, such as neural networks, consider F and G as black box functions. However, one could also model them as differentiable directed graphs that are dynamically composed by functional blocks and rely on general-purpose Automatic Differentiation (AD) [3]. AD enables to compute derivatives through accumulation of numerical values during code execution to generate numerical derivative evaluations as opposed to derivative expressions. This class of techniques performs these operations by using symbolic rules of differentiation but keeps track of derivative values as opposed to the resulting expressions. By extending the arithmetic of AD to higher order derivatives of multivariate functions, all numerical computations appear as compositions of a finite set of elementary operations for which derivatives are known (instead of involving numerical differentiation). Next, by combining the derivatives of the constituent operations through the higher order chain rule one obtains the derivative of the overall composition. Observe that fixing the highest derivatives order (instead of considering it as a free parameter) limits the dimensionality of the search space. It remains an open question whether imposing such constraint could improve the computation time vs. prediction accuracy tradeoff.

A second neural network can then be associated that i) controls the dynamic composition of the differentiable graph and ii) is trained to predict the numerical values corresponding to the functions F and G defined by (1). For this purpose:

- Multi-dimensional neural networks also referred to as Clifford Neural Networks (CNN) [1] [6] can be considered for training each (group of) vertex to identify the operation each of them performs on input features/activations. This class of networks can model multi-level interactions: within data points

(between vector components) and between data points (between vectors) while interactions themselves take a vectorial form (instead of a scalar as in real-valued neural networks).
– Combining both group- and exclusivity- sparsity methods shall be enforced during the training phase to exploit positive and negative correlations between features.
– Techniques to interrogate the model to determine why and what has been learned shall then be elaborated in order to i) enable interpreting the learned model and its associated features and ii) translate this information into composition primitives.

Further, the predictive power of the learned model shall provide capability beyond single-step prediction, i.e., given the value of a variable/quantity at time t, predict its value at time $t + 1$, but instead enable for multi-step predictions over longer time horizon $t + k$, $k \gg 1$. Involving multiple steps allows in turn to incorporate memory effects in learning the temporal dynamics and cover problems with a non-Markovian dynamical structure. As available training data may not be invariant to time shifts, this non-stationarity property would be better captured by the modeling capacity of recurrent neural networks (RNN).

– In the discrete time domain, we could consider the training of multi-directional recurrent neural networks (MD-RNN) [10] for this purpose since training data are often characterized by more than one spatio-temporal dimension. Instead of pre-processing the data to one dimension and involving a single recurrent connection, MD-RNNs uses as many recurrent connections as there are dimensions in the data.
– In the continuous time domain, their combination with continuous time RNN (CT-RNN) [9] would enable handling training over continuous timescales while increasing the computational efficiency compared to standard discrete-time RNNs. The latter induce dependence of the resulting models on the sampling period used in the learning process when no information is given about the model trajectories between the sampling instants.

5 Discussion

Extracting physical laws and models from data involves to a broad set of learning tasks with increasing level of complexity. When the domain is known, the task can be seen as a PDE model fitting combining a neural network that approximates the PDE solution and another that checks the concordance between the approximated solution and the fitted/trialed model. The latter differentiates the numerical solution following the known analytic form of the PDE. One approximates a physical model from a relatively well delimited hypothesis space and in the simplest case only parameters needs to be identified. Unfortunately, this brings us back to the main observation that neural-based learning is (still) not an essential modeling tool but mainly a pre-existing model fitting tool.

When the domain is unknown, discovering the closed-form expression of the PD(A)E model even with algorithmic methods remains highly challenging. This paper by detailing the two main machine learning-based approaches (dictionary-based learning and a particular variant of the previous model) shows that current neural network training algorithms but also neural network model selection are not (yet) at the level required to perform such tasks. Of course, we could still decompose this case into situations where the physical model can't be appropriately identified (hidden) but is known and the case where the physical process itself has never been identified. However, the main problem remains: the hypothesis/pattern search and matching process implemented in the neural learning paradigm is still too limited to cope with the exploration a large spaces in particular when multi-scale interactions among input variables are involved.

References

1. P. Arena, L. Fortuna, G. Muscato and M.G. Xibilia, Neural networks in multidimensional domains, Lecture Notes in Control and Information Sciences (LNCIS), vol.234, Springer-Verlag, 1998.
2. I. Babuska, and B.Q. Guo, The h, p and h-p version of the finite element method: basis theory and applications, Advances in Eng. Software, 15(3-4):159-174, 1992.
3. A.G. Baydin, B.A. Pearlmutter, A.A. Radul, and J.M. Siskind, Automatic differentiation in machine learning: a survey, JMLR, 18:1-43, 2018.
4. H.G. Bock, Recent advances in parameter identification techniques for ODE, Numerical treatment of inverse problems in differential and integral equations, Springer, 1983, pp.95-121.
5. L. Breiman, Statistical Modeling: The Two Cultures, Statistical Science, 16(3):199-231, 2001.
6. S. Buchholz, G. Sommer, On Clifford neurons and Clifford multi-layer perceptrons, Neural Networks, 21:925-935, 2008.
7. N. Baker et al., Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence, Feb. 2019. Available at https://www.osti.gov/biblio/
8. H. Edelsbrunner, D. Letscher, and A.J. Zomorodian, Topological persistence and simplification, Discrete & Computational Geometry, 28(4):511-533, 2002.
9. K.L. Funahashi, and Y. Nakamura, Approximation of Dynamical Systems by Continuous time Recurrent Neural Networks, Neural Networks, 6:183-192, 1993.
10. A. Graves, S. Fernandez, J. Schmidhuber, Multi-Dimensional Recurrent Neural Networks, Proceedings of ICANN 2007, Part I, LNCS 4668, pp.549-558, Springer Berlin Heidelberg, 2007.
11. J. Hadamard, Sur les problèmes aux dérivées partielles et leur signification physique, Princeton University Bulletin, pp.49-52, 1902.
12. K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks, 4(2):251-257, 1991.
13. Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.
14. A. Ivakhnenko, Polynomial theory of complex systems, IEEE Transactions on Systems, Man and Cybernetics, 1(4):364-378, 1971.

15. B.O. Koopman, Hamiltonian systems and transformation in Hilbert space, Proceedings of National Academy of Sciences (PNAS) of the United States of America, 17(5):315-318, 1931.
16. D.Marcek and M. Marcek, Neural Networks and their Applications, EDIS Publications, Slovakia, 2006.
17. I Markovsky, Low-Rank Approximation, Algorithms, Implementation, Applications, 2018, Springer.
18. A. Mauroy, and J. Goncalves, Koopman-based lifting techniques for nonlinear system identification, https://arxiv.org/pdf/1709.02003.pdf.
19. S.K. Oh, W. Pedrycz and B.J Park, Polynomial neural networks architecture: Analysis and design, Computational Electrical Engineering, 29:703-725, 2003.
20. J.F. Price, Polynomial differential equations compute all real computable functions on computable compact intervals, American Journal of Physics, 71:437-447, 2003.
21. J.A.A. Opschoor, P.C. Petersen, and C. Schwab, Deep ReLU Networks and High-Order Finite Element Methods, Research Report No.2019-17, Seminar for Applied Mathematics (SAM), ETH Zurich, January 2019
22. M. Raissy, Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, Journal of Machine Learning Research, vol.19, pp.1-24, 2018.
23. S.H. Rudy, S.L. Brunton, J.L. Proctor, and J.N. Kutz, Data-driven discovery of partial differential equations, Science Advances, 3(4), 2017.
24. J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations, Journal of Comp. Physics, 375:1339–1364, 2018.
25. V. Vemuri, Inverse Problems, in (Eds. and G. A. Bekey and B. Y. Kogan), Modeling and Simulation: Theory and Practice, A Memorial Volume for prof. Walter J. Karplus, pages 89-101, Kluwer Academic Publishers, Boston, 2002
26. A. Zomorodian, and G. Carlsson, Computing persistent homology, Discrete & Computational Geometry, 33(2):249-274, 2005.
27. L. Zjavka, Polynomial neural network, Proceedings of 7th European Conference Information and Communication Technologies, pp.277-280, June 2007.
28. L. Zjavka, Differential Polynomial Neural Network, Journal of Artificial Intelligence, 4:89-99, 2011.