Springer Nature: Computer Science manuscript No.
(will be inserted by the editor)

Exactly predictable functions for simple neural networks

Jack Yost · Luis Rizo · Xing Fang · Qichang Su · Rainer Grobe

Received: date / Accepted: date

Abstract We examine the degree of accuracy of simple feedforward neural nets with N inputs and a single output to forecast time series that represent analytical functions. We show that the subspace of those functions, whose higher order derivatives can be clustered into a finite number of linearly dependent groups, can be forecasted exactly by a neural net. Furthermore, we derive generally applicable summation and product rules that permit us to calculate the associated optimum connection weights for the particular network architecture for complicated but exactly predictable functions. If a general network is initialized with these particular weights, the learning process for general data (with noise) can be significantly accelerated and the forecasting accuracy increased. We also show that neural nets can be used to predict the finite value of diverging sums, which is a generic problem for most perturbation-based approaches to physical systems.

Keywords Exactly predictable functions · Simple neural networks · Time series forecast · Summation and product rules

1 Introduction

Among the many computational benefits of artificial neural networks, the capability to predict the future value of a time series from past information is of special interest to the physical sciences [1]. It has been suggested that these networks can be superior to traditional statistical forecast approaches [2]. In classical supervised machine learning [3], a neural net is often considered as

J. Yost (jmyost1@ilstu.edu), L. Rizo (lmrizo1@ilstu.edu), Q. Su (qcsu@ilstu.edu), and R. Grobe (grobe@ilstu.edu)
Intense Laser Physics Theory Unit and Department of Physics Illinois State University, Normal, IL 61790-4560, USA
X. Fang, xfang13@ilstu.edu, ORCID: 0000-0001-8574-9149, tel: 309-438-8133
School of Information Technology Illinois State University, Normal, IL 61790-5150, USA

a black box process where N inputs, typically given by historical values of a sampled function $f(t_n)$ with $t_n = t - nh$ and $n = 1, 2, ..., N$ and step size h, can be successfully trained to predict the future value $f(t)$.

Artificial neural networks have a rich history with early suggestions dating back to 1890 [4] when James defined a neural process of learning. In 1943 McCulloch and Pitts [5] introduced first mathematical models and in 1954 Hebb and his collaborators [6] provided first simulations. In 1958 Rosenblatt [7] introduced the perceptron, which in 1969 [8] was shown to have some severe limitations. In the mid 1980's the new concepts of back-propagation [9] and multi-layer networks [10] lead to some re-emergence of the field leading now to today's central role in machine learning. Due to its widespread applications in science and technology, there have been many reviews, books and journals published in this field. The system we examine in this work is closest related to the formalism used in ARIMA-like (autoregressive integrated moving average) time series calculations [11–15]. This model is frequently used in statistics, economics, and finance to better understand the data or to predict future points in the series.

To have a simple example of linear regression [16], a network with $N = 3$ inputs can be trained with numerous training pairs, where each pair (characterized by a specifically given time t) consists of three numerical values that sample the unknown function $[f(t - 3h), f(t - 2h), f(t - h)]$ and an associated target value $f(t)$. The quality of a particular network architecture is then measured by its ability to generalize from its training set to previously unseen regions, i.e. to a new time t in the future. We assume here that the measured data originate from an unknown but analytical function $f(t)$. For several applications, it can be sufficient to just predict a single future value $f(t)$ from its historical data. In many situations, where data further into the future need to be predicted, the exact (measured) value of $f(t)$ is then used to predict $f(t+h)$. This contrasts more powerful schemes, where the actually predicted value for $f(t)$ is used to consecutively predict $f(t+h)$ and so on. For the latter iterative scheme, the accuracy of the first predicted value for $f(t)$ is therefore crucial to guarantee the quality of all of its consecutive predictions $f(t + nh)$. In all of these contexts, the questions about the optimal hyperparameters (such as the number of inputs and length and width of hidden layers) or the possibility of over- or under-fitting are all important challenges of eminent practical relevance.

In order to better understand the mechanisms on which efficient neural nets are based on, we have examined the following simple questions. For a given net's architecture with a finite number of N inputs, are there specific sets of functions, for which the future value $f(t)$ can be actually predicted from the past values $f(t_n)$ without any error? If these exists, can one construct the corresponding associated sets of optimal connection weight factors? If those exactly predictable functions (which we abbreviate by epfs) exist, are the sets of new functions obtained from the sums (or even the products) of those original epfs again exactly predictable?

An important breakthrough was obtained by Hornik et al. [17,18] and others [19–21] who suggested that a neural network (even with just one input and output) can approximate any arbitrary function, where its accuracy is determined by the number of units in the hidden layer. This theorem is complementary to our findings where we show that a single neuron with N inputs and a single output can also approximate arbitrary functions. Here the accuracy is improved by increasing the number of input units.

The contribution of this article is two-fold. We show that there exists indeed a set of elementary exactly predictable functions (epfs) and determine their associated optimum weights. We prove that any sums and even products of these functions are epfs again, but these require a larger number of input channels. We show that neural nets can be used to compute diverging series. Furthermore, the weights factors of epfs can be used to enhance the learning efficiencies of networks.

This article is structured as follows. In Section 2 we provide the underlying theory for epfs based on the observation that in this case all derivatives can be partitioned into a finite number of linearly dependent clusters. We also introduce the superposition principle for sums and products of epfs and provide the rules to determine their weight factors of complicated functions in terms of those associated with more elementary functions. Section 3 is devoted to two applications. We show how that the weights of epfs can be used to accelerate the learning rate. The network can also be used to evaluate diverging sums. Section 4 generalizes epf to multineuron networks. In Section 5 we provide an outlook to open questions and future challenges.

2 Elementary and composite exactly predictable functions

2.1 General theory

It is obvious that a function $f(t)$ can never be uniquely identified from only a finite number of sampled data $f(t_n)$ with $t_n = t - nh$ for $n = 1, 2, ..., N$. Therefore, the prediction of the future value of the underlying function f at time t (corresponding to n=0) based on just a finite amount of historical information $f(t - nh)$ is not a problem that has a mathematically unique solution. However, as we will discuss in this work, there exists an interesting subspace of functions, for which a *finite* amount of historical data $f(t - nh)$ is actually sufficient to predict their future value exactly at any time.

To begin with the simplest possible system, we examine a single neuron with N inputs, characterized by N weight factors W_n, which is similar to the ARIMA (autoregressive integrated moving average) configuration. Furthermore, we neglect any bias or activation function such that it maps the N inputs, denoted by past values of the (unknown) $f(t_n) \equiv f(t - nh)$ to a single output, i.e., $y_{out} = \sum_{n=1}^{N} W_n f(t - nh)$. We assume here that the unknown function f(t) is real, analytical and can be Taylor expanded, i.e., $f(t - nh) = \sum_{m=0}^{\infty} a_m (nh) f^{(m)}(t)$, where $f^{(m)}(t)$ denotes the m-th derivative

of $f(t)$ with $f^0 = f(t)$ and where $a_m(nh) \equiv (-nh)^m/m!$ We therefore obtain $y_{out} = \sum_{n=1}^{N} W_n f(t-nh) = \sum_{n=1}^{N} W_n \sum_{m=0}^{\infty} a_m(nh) f^{(m)}(t)$. The key question is if we can construct a *finite* set of weight factors W_n, such that y_{out} matches exactly $f(t)$, i.e., we require

$$f(t) = \sum_{n=1}^{N} W_n \sum_{m=0}^{\infty} a_m(nh) f^{(m)}(t) \tag{1}$$

to be valid for all times t. In the general case, where all derivatives $f^{(m)}(t)$ are linearly independent of each other, the equality of the left and right hand side of Eq.(1) would require that we have

$$\sum_{n=1}^{N} W_n a_m(nh) = \frac{(-h)^m}{m!} \sum_{n=1}^{N} W_n n^m = \delta_{m,0} \tag{2}$$

for all integers m and where $\delta_{m,0}$ denotes the Kronecker delta symbol. This an infinite set of equations with N unknowns W_n that -for any general function f(t)- can only be satisfied for $N = \infty$ coefficients W_n.

This feature should not be confused with the universal approximation theorem [17–21] for networks, which showed that any measurable function $f(t)$ can be represented arbitrarily well by a standard multilayer feedforward configuration. In contrast to our (single-neuron) architecture where the desired degree of accuracy grows with the number N of input units, here the accuracy grows with the number of units (the width) of the hidden layer.

There is a special subset of functions $f(t)$, which we will denote as *e*xactly *p*redictable *f*unctions (or epfs), where not all of the derivatives $f^{(m)}(t)$ are independent of each other. In this case, the derivatives can be grouped into a *finite* set of linear-dependent classes, that we denote by $g_\alpha(t)$. In other words, for any time t, each derivative can be assigned uniquely to an integer index $\alpha(m) = 0, 1, 2, 3, ..., N_\alpha - 1$ such that $f^{(m)}(t) = k_m g_{\alpha(m)}(t)$, where k_m is an irrelevant proportionality constant that does not depend on time. If there is a total number of N_α (derivative) clusters, then the Taylor expansion of Eq. (1) can be conveniently regrouped as

$$f(t) = k_0 g_{\alpha(0)}(t) = \sum_{n=1}^{N} W_n \sum_{m=0}^{\infty} a_m(nh) k_m g_{\alpha(m)}(t) \tag{3}$$

The second summation in Eq. (3) over all integers m can be partitioned into the summation over those groups that are characterized by the same value for α as

$$\sum_{m=0}^{\infty} a_m(nh) k_m g_{\alpha(m)}(t) = \sum_{\alpha=0}^{N_\alpha - 1} g_\alpha(t) \sum_{m'(\alpha)}^{\infty} a_{m'(\alpha)}(nh) k_{m'(\alpha)} \tag{4}$$

In other words, we obtain

$$k_0 g_{\alpha(0)}(t) = \sum_{n=1}^{N} W_n \sum_{\alpha=0}^{N_\alpha - 1} g_\alpha(t) \sum_{m'(\alpha)}^{\infty} a_{m'(\alpha)}(nh) k_{m'(\alpha)} \tag{5}$$

Here m' in the m' summation combines all m in the m summation that give rise to the same α. To make this equation valid for all times t, only the (time-independent) pre-factors of the (mutually independent) $g_{\alpha(m)}(t)$ have to vanish. This means that the weights W_n are uniquely specified as solutions to the following set of N_α linear equations, assuming $\alpha(m=0)=0$ and $\alpha \leq m$

$$\begin{aligned} for\ \alpha = 0, \quad k_0 &= \sum_{n=1}^{N} W_n \Big(\sum_{m'(0)}^{\infty} a_{m'(0)}(nh) k_{m'(0)} \Big) \\ for\ \alpha \neq 0, \quad 0 &= \sum_{n=1}^{N} W_n \Big(\sum_{m'(\alpha)}^{\infty} a_{m'(\alpha)}(nh) k_{m'(\alpha)} \Big) \end{aligned} \tag{6}$$

where the summation $\sum_{m'(\alpha)}^{\infty}$ extend only over those subgroups of integers, for which $\alpha(m') = \alpha$ takes the same value. This means, if there are N_α (independent) derivative clusters, then we require only at most $N = N_\alpha$ weight factors to construct a unique set of weights W_n. We denote with N the *class* of the epf.

For small numbers N_α, the resulting elementary epfs can take a very simple form. For example, a class $N_\alpha = 1$ epf is the exponential function $f(t) = a\exp(bt)$, with arbitrary constants a and b. In this case, all derivatives are linearly dependent of each other and we have $W_1 = \exp(bh)$, and all other weights vanish. Similarly, for $N_\alpha = 2$, all derivatives of the function $f(t) = a\sin(bt)$ can be grouped into only two clusters, leading to the weight factors $W_1 = 2\cos(bh)$ and $W_2 = -1$.

Another interesting example are monomials $f(t) = at^L$, which lead to $N_\alpha = L+1$ clusters, reflecting the fact that all $f^{(m)}(t)$ for $m > L$ vanish. Here the optimum weight factors W_n are given by binomial coefficients (L+1,n) with alternating signs, i.e., $W_n = (-1)^{n+1}(L+1, n) \equiv B_{n,L+1}$. In contrast to the class=1 or =2 epfs such as $\exp(at)$ or $\sin(at)$, these weight factors do not dependent on the numerical value of the temporal grid spacing h. These particular weight factors will play an important role in accelerating the learning speed for neural nets as we will suggest in Section 3.1.

In Table 1 we have summarized some of the elementary class=3 and class=4 epfs together with their weight factors W_n. We see that the weights do not depend on the pre-factors α and β.

The possibility of a function $f(t)$ to be decomposable into a finite number of derivative clusters is also a direct consequence of the fact, that each epf is also a solution to a corresponding autonomous linear differential equation. Here the required functional relationship among certain groups of derivatives is explicitly stated. If a function $f(t)$ is a solution to the general N-th order differential equation $d^N f/dt^N = \sum_{n=1}^{N} c_n d^n f/dt^n$ with arbitrary time-independent

Table 1 Examples of some class=3 and class=4 exactly predictable functions $f(t)$ together with their perfect weight factors W_n such that $f(t) = \sum_{n=1}^{N} W_n f(t-nh)$ is exact for $N = 3$ or $N = 4$.

epf $f(t)$	W_1	W_2	W_3	W_4
$\alpha t^2 e^{at}$	$3e^{ah}$	$-3e^{2ah}$	e^{3ah}	0
$\alpha + \beta \sin(at)$	$1 + 2\cos(ah)$	$-1 - 2\cos(ah)$	1	0
$\alpha \cos^2(at) e^{bt}$	$e^{bh}[1 + 2\cos(2ah)]$	$-e^{2bh}[1 + 2\cos(2ah)]$	e^{3bh}	0
$\alpha t \sin(at)$	$4\cos(ah)$	$-4 - 2\cos(2ah)$	$4\cos(ah)$	-1
$\alpha t + \beta \sin(at)$	$2 + 2\cos(ah)$	$-2 - 4\cos(2ah)$	$2 + 2\cos(ah)$	-1

constants c_n then $f(t)$ is automatically an epf of class=N and vise versa. The relationship between discrete time-series analysis and differential equations has been addressed in [22–24]. The specific functional form of $f(t)$ follows from the multiplicity of the roots of the corresponding characteristic polynomial [24]. One might attempt to relate the N coefficients an of the differential equation to the corresponding perfect weight factors W_n for the N-point predictive scheme for $f(t)$ based on $f(t - nh)$, but this is not so trivial, as the W_n depend in general nontrivially on the chosen grid spacing h.

On the other hand, for the special case, where the N coefficients are chosen as the alternating binomials $a_n = B_{n,N}$, the differential equation has the N superpositions of product terms $t^j \exp(t)$ with $j = 0, 1, ..., N-1$ as its solution. To return to some of the concrete examples of Table 1, the class=2 function $f(t) = t \exp(at)$ fulfills $d^2f/dt^2 = 2a\ df/dt - a^2 f$. The class=3 function $f(t) = \alpha t^2 \exp(at)$ is a solution to the third-order differential equation $d^3f/dt^3 = 3a\ d^2f/dt^2 - 3a^2\ df/dt + a^3 f$ and class=4 given by $f(t) = \alpha t + b \sin(at)$ is a solution to the fourth-order differential equation $df^4/dt^4 = -a^2\ df^2/dt^2$

2.2 The general superposition principle for summations of epfs

One could (incorrectly) conjecture that the set of exactly predictable functions is limited to only a small group of those few functions, that have a trivial dependence on time such as monomials, exponentials and sine functions as sketched in Table 1. However, this is not true. It turns out that the sum of any two epfs $f(t)$ and $g(t)$ is automatically an epf again. More precisely, one can show that if the two functions $f(t)$ and $g(t)$ are of class N and M, then the new function $F(t) \equiv f(t) + g(t)$ is at most of class $N + M$. The proof for this claim is straight forward and follows directly from Eq. (6) when applied to $F(t) = g(t) + f(t)$. If the derivatives of $f(t)$ fall into N clusters and those of $g(t)$ into M clusters, then -due to the linearity of Eqs. (6)- we can regroup the derivatives of $F(t)$ into (at most) $N + M$ clusters. As a result, we would have to solve $(N + M)$ equations for the weight factors of $F(t)$.

Furthermore, it is even possible to derive a general relationship between the sets of perfect weights associated with $f(t)$ [denoted here by $W_n(f)$ with $n = 1, 2, 3, ..., N$] and those of $g(t)$ [denoted by $W_m(g)$ with $m = 1, 2, ..., M$]

and the resulting perfect weights for the sum $F(t)$, denoted by $W_k(f+g)$ with $k = 1, 2, ..., N+M$. One finds the linear superposition law for the $k-th$ weight

$$W_k(f+g) = W_k(f) + W_k(g) - \sum_{n=1}^{N} W_n(f) W_{k-n}(g) \tag{7}$$

with $W_k(f) = 0$ for $k \leq 0$ and $k > n$ and with $W_k(g) = 0$ for $k \leq 0$ and $k > M$. This general superposition law for the sums is one of the main findings of this article. A tedious but straightforward proof for this general law is derived in Appendix A. While the proof involves higher-order products of the underlying weight factors $W_n(f)$ and $W_m(g)$, it is quite remarkable that the final expressions for $W_k(f+g)$ require only linear and bi-linear combinations of the original weights.

An important consecutive application of the sum rule for monomials $f(t) = a\ t^L$ with increasing order L leads to the important fact that also general polynomials of any order N are epfs. However, as in this particular case the derivatives of the lower order monomials are naturally linearly dependent on the derivative of higher order monomials, their class (=number of required inputs for an exact predictability) is simply given by that of the associated monomial of the highest order. Even more remarkable, the associated perfect weights for the polynomial $f(t)$ are completely identical to the ones associated with this particular monomial. In the prior Section 2.1 we have shown that its weight factors are given by

$$W_n = (-1)^{n+1}(L+1, n) \tag{8}$$

for $n = 1, 2, 3, ..L+1$, again independent of the numerical value of the temporal grid spacing h.

2.3 The general superposition principle for products of epfs

Quite remarkable, the possibility of the clusterability of higher-order derivatives applies even to the products of epfs. In other words, if each of two functions $f(t)$ and $g(t)$ is an epf (of class N and M), then one can show that even the new function given by the product of $F(t) \equiv f(t)g(t)$ is again an epf and therefore can be exactly forecasted by the net. The proof follows directly from the Leibniz product rule for the n-th derivative for $F(t)$, where $F^{(n)}(t) = \sum_{k=0}^{n}(n,k) f^{(n-k)}(t) g^{(k)}(t)$ with the binomial coefficient (n,k). If the derivatives of $f^{(n-k)}(t)$ can be clustered into N groups of linearly dependent functions and, similarly, $g^{(k)}$ into M groups, then it is obvious that also the derivatives of $F(t)$ can be clustered into at most NM clusters. In other words, $F(t)$ is exactly predictable as well. In this sense, the superposition principle for sums can be generalized to products as well.

While for the case of summations it was possible to provide general construction rules of the perfect weight factors from those of the underlying two functions $f(t)$ and $g(t)$, see Eq. (7), in the case of products these rules cannot

be found for general N and M and the resulting expressions are extremely complicated. In Appendix B we derive the expressions for the 9 weight factors associated with the product $F(t) = f(t)g(t)$, where $f(t)$ and $g(t)$ are class=3 epfs.

2.4 Concrete numerical illustration and sensitivity to noise

To illustrate the sensitivity of the predictive power of the optimum weights, we have examined the test function $f(t)$=$3\exp(a_1 t)\cos(b_1 t) + \exp(a_2 t)\cos(b_2 t)$, which is a class=4 epf. In other words, the knowledge of this function at just 4 past times $f(t-nh)$, $n = 1, 2, 3, 4$ together with the perfect four weights W_n is fully sufficient to predict $f(t)$ exactly. The predicted value for $f(t)$ can then be used iteratively to forecast $f(t+h)$ and then $f(t+2h)$, etc.

To have a concrete numerical example, we chose the specific parameters ($a_1 = -2, b_1 = 70, a_2 = 2$ and $b_2 = 20$). While being functionally simple for illustrative purposes, $f(t)$ still has sufficient complexity that it is very difficult to humanly predict its future behavior for $0.5 \leq t$ from the known historical data $[t, f(t)]$ for $0 < t < 0.5$ by simple inspection of the graph. In the inset of Figure 1 we display its complicated behavior.

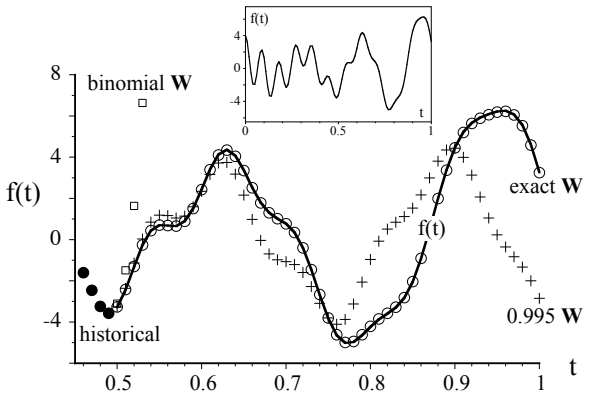

Fig. 1 The predictions of the future 51 values of the function $f(t)$ based solely on its four historical values at times $t = 0.46, 0.47, 0.48$ and 0.49. The future times are $t = 0.5 + (n-1)0.01$ with $n = 1, 2, ..51$. The second set of predictions was based on a set of four imperfect weights W_n, obtained by a reduction of 0.5% of the optimum weights. The third set of predictions simply used a standard cubic extrapolation scheme, here we display (with the squares) only the first four predictions, as these data quickly grow out of bounds. The test function was chosen as $f(t) = 3\exp(-2t)\cos(70t) + \exp(2t)\cos(20t)$ as shown in the inset for the range $0 \leq t \leq 1$.

In Figure 1 we show how the four historical values of $f(t)$ at past times $t = 0.46, 0.47, 0.48$ and 0.49 (solid circles) can be used to predict consecutively the 51 future values of $f(t)$ (open circles) at times $0.5 + (n-1)0.01$ with $n = 1, 2, ..., 51$. The match between the forecasted and the true value $f(t)$ is perfect, as the four weight factors are chosen as $\mathbf{W}(h) = \mathbf{W}(0.01) = (3.50, -5.00, 3.48, -1.00)$. The analytical form of these weights for arbitrary a_1, b_1, a_2, b_2 and h is derived in Appendix A. While due to the four different growth, decay and periodic time scales associated with each of the four constituent functions (proportional to $a_1^{-1}, b_1^{-1}, a_1^{-1}$ and b_2^{-1}) this test function is sufficiently complicated, its entire future behavior is nevertheless accurately predicted by only 4 historical data points.

The second set of predictions shown in the same figure is based on slightly inaccurate weight factors. These were obtained by reducing the optimum weights by half a percentage each, given by $\mathbf{W}_{new} = 0.995\mathbf{W}$. While the first five future points are still predicted sufficiently accurate, we see major deviations for longer times, where the inaccuracies accumulate as expected due to the consecutive nature of this forecasting scheme. This suggests that the accuracy of the prediction depends rather sensitively on the quality of the weight factors. This high sensitivity for any long-time forecasting is not so unexpected. Due to the consecutive application of the same weight factors for each predicted value, these predictions scale highly nonlinearly with the weights W_n. For example, the 51^{th} prediction $f(t)$ for the long time $t = 1[= 0.5+(n-1)0.01$ with $n = 51$, based on the historical data from $t < 0.5]$ scales like W_1^{51}, which is obviously rather sensitive to the accuracy of W_1 itself.

For a comparison with the quality of more standard extrapolation approaches [16], we have also displayed (by the open squares) the predictions of the cubic polynomial based scheme. Here the fitted cubic polynomial of the form $P(t) = -6050 + 38973\, t - 83522\, t^2 + 59529\, t^3$ was obtained from the four historical values $f(0.46)$, $f(0.47)$, $f(0.48)$ and $f(0.49)$. However, the first four predicted values $P(0.5)$, $P(0.51)$, $P(0.52)$ and $P(0.53)$ vary significantly from the corresponding true future values of $f(t)$. This disagreement is, of course, expected as any polynomials of $N-th$ order can take at most $(N-1)$ maxima and therefore can never approximate any oscillatory function as tested in this case. The same predictions could have also been obtained if the data were computed not from $P(t)$ (as an intermediate step) but directly from the four binomial weight factors $\mathbf{W} = (4, -6, 4, -1)$. These weight factors assume that the underlying function that matches the historical data is a cubic polynomial and make the forecasting accordingly.

In closing, we want to stress again that all of the predictions for $t \geq 0.5$ in this article were solely based on only four historical data. There are, of course, many applications (for example in predictions for financial markets) where a single-time step prediction is sufficient. Here the weight factors can be constantly relearned in each step and -in contrast to our work- the prior four values are not the predicted values themselves, but actual historical measured data. As these truly short time predictions can self-correct as time evolves, they are much more easily obtained. For example, had we used in each step

the true four values given by $f(t)$, then using the simple binomial weights $\mathbf{W} = (4, -6, 4, -1)$ would have led to predictions that are graphically indistinguishable from the true graph of $f(t)$ for the entire range $t \geq 0.5$.

3 Applications

3.1 Increased speed of convergence for smartly initialized networks

A crucially important aspect of any network is its efficiency with which it can learn from the training set. In many situations, the efficiency of the learning process is determined by the quality of the initial choice of the weights. While there has been some guidance in the literature [13] about desirable magnitude ranges for the initial weights, they are often assumed to be random. In this section, we will suggest that the efficiency of the learning process can be significantly increased if in the initialization of the net the alternating binomial weights given by $B_{nN} = (-1)^{n+1}(N, n)$ for $n = 1, 2, 3..N$ are used.

In Section 2.2 we have derived the set of N perfect connection weights W_n that would make the forecasting exact if the function to be forecasted is a polynomial of degree $N-1$. We have shown that this set given by $W_n = B_{nN}$ is independent of the numerical value of the temporal grid spacing h. This means if the net's forecasting of $f(t)$ is based on this particular set of weights, it would assume that the N points associated with the historical data [obtain by sampled values $f(t - nh)$] describe actually a $(N-1)^{th}$ order polynomial and make the prediction for $f(t)$ accordingly.

Let us assume we consider here an algorithm that tries to predict the future value of the function $f(t)$ from the N historical data $f(t-nh)$ $(n = 1, 2, ..., N)$. Furthermore, let us assume the function is sampled over a total of J temporal grid points $t_j = (j-1)h$, $j = 1, 2, ..., J$, then we would have a maximum number of $N - J$ training sets for the learning process available.

For the simplest case of a threshold-free single neuron, the accuracy depends of course on the deviation of the predicted future value, $f_{pred}(t) = \sum_{n=1}^{N} W_n f(t - nh)$ and the true future value, given by $f(t)$. For a given set of weight factors W_n, the cost (loss function) associated with the entire training sets is often defined by the mean squared distance $E(\mathbf{W}) \equiv J^{-1} \sum_{j=1}^{J} [f(t_j) - \sum_{n=1}^{N} W_n f(t_j - nh)]^2$. This loss function plays a dual role, first, when evaluated over the historical times, it guides the learning process based on the historical values (via back propagation of the error) as an indicator of the learning progress, and second, when summed over future grid points, it can act as a criterion for the predictive power of the finally obtained weights. Due to the simple quadratic dependence of the error E as a function of the weights, there is at most a single (i.e. global) minimum in this N-dimensional space spanned by the weights. For the special case of epf functions with N weights, this minimal error is identical to E=0.

This means that any gradient based optimization algorithm for the optimal weights will converge to the global minimum. However, this minimum might

be very shallow and the learning process usually stops after a certain given error tolerance is reached. In this sense it might be from a practical point of view non-trivial to converge to the precise global minimum in finite CPU time. There can be also a sub surface along which the minimum lies. For example, this later situation can occur if a class=N epf is being optimized with more than N weight factors.

To provide a concrete illustration, let us consider again the test function $f(t) = \exp(a_1 t)\cos(b_1)t + \exp(a_2 t)\cos(b_2)t$ from Section 2.4. This function is an epf of class $N = 4$ and in Appendix C we have derived the analytical form of its associated four weights W_n. It follows that in the limit of very small grid spacings $h \to 0$, the weights approach the binomial forms, i.e. $W_n \to B_{n4}$. This limiting behavior is crucially important. It makes the binomial coefficients rather universally applicable to accelerate the learning scheme for any function $f(t)$ if only the step size h is chosen sufficiently small. The h-dependence of the true weights is also graphed in the Appendix.

In our numerical illustration, we use $J = 50$ linearly spaced historical times $t_j \equiv (j-1)h$ ranging from $t_1 = 0$ to $t_J = 0.5 - h$ with a grid spacing of $h \equiv 0.5/J$. Therefore, the J data pairs $[t_j, f(t_j)]$ provide the 46 training sets. In Figure 2a we have graphed the progress of the learning process as a function of the number of epochs. Each epoch consists here of 46 training sets, each given by $[f(t_{j-4}), f(t_{j-3}), f(t_{j-2}), f(t_{j-1})] \to f(t_j)$, for $j = 5, 6, ..., 50$. The specific learning algorithm for the weights was based on the stochastic gradient decent (SGD) method. We used 0.1 for the initial learning rate. For comparison, in a separate study, we also used the standard gradient-based adaptive learning rate optimization algorithm (ADAM) [26–30] and found similar results [31]. For the initialization of the net we used the parameters $\mathbf{W} = (-5, 6, -8, 10)$. In Figure 2b we display the average error given by $E(\mathbf{W}) \equiv J^{-1}\sum_{j=1}^{J}[f(t_j) - \sum_{n=1}^{4} W_n f(t_j - nh)]^2$ that compares the predicted value $\sum_{n=1}^{4} W_n f(t_j - nh)$ with the target value $f(t_j)$ for the training set.

We see that with a random initialization of the four weights, the learning algorithm requires about 70,000 epochs for the weight factors to converge. The values of the converged weights match those perfect weights $\mathbf{W}(0.01) = (3.50, -5.00, 3.48, -1.00)$, as discussed in Section 2.4 above.

In Figure 2b we have displayed the loss function for the same learning process, however, here we have initialized the weights as $\mathbf{W} = (4, -6, 4, -1)$. We see a significant enhancement of the learning process. The weights converge to the same final set, but the rate of convergence is significantly enhanced. Here we obtain weights that match the exact one by an error of less than $1.65 \times 10^{-7}\%$ already after 47400 epochs.

In order to suggest the universal applicability of these binomial weights to enhance the learning process, we have used a test function that is not an epf for any N-point scheme. Here we choose $g(t) = 3\exp(a_1 t)\cos(b_1 t) + \exp(a_2 t^4)\cos(b_2 t)$, where we have simply replaced the second exponential in $f(t)$ by a super Gaussian. For easier comparison, we have kept all the parameters the same. This case differs from the above example by only two minor

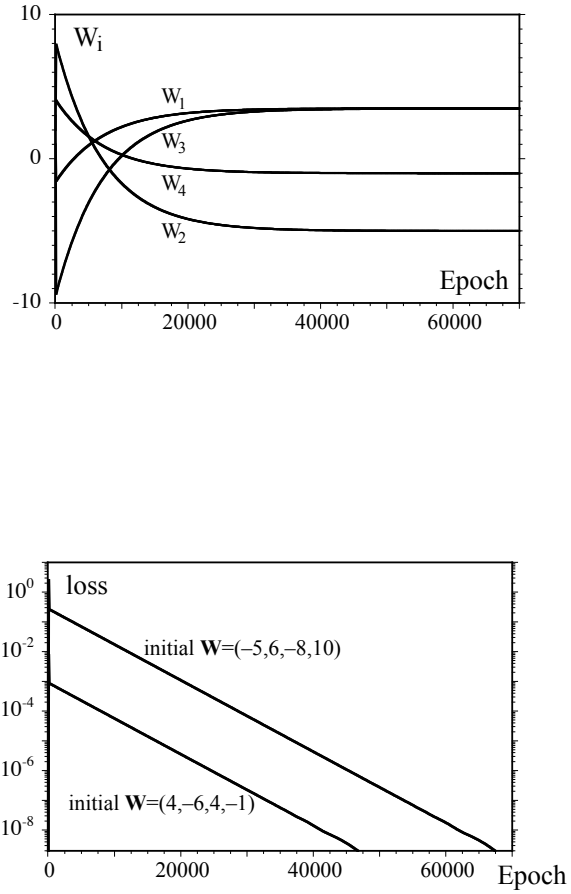

Fig. 2 (a) The (SGD-based) learning process of the four weights W_n for the test function as a function of the number of epochs based on 46 training sets. After about 70,000 epochs the error between the exact perfect weights and the learned ones is less than $1.5810 \times 7\%$. Here the initialization was based on $\mathbf{W} = (-5, 6, -8, 10)$; (b) The decrease of the loss function as a number of the epochs for the random initialization $\mathbf{W} = (-5, 6, -8, 10)$ and also for the initialization based on the binomials weights $\mathbf{W} = (4, -6, 4, -1)$. The test function was chosen as $f(t) = 3\exp(-2t)\cos(70t) + \exp(2t)\cos(20t)$ as shown in the inset of Fig. 1 for the range $0 \leq t \leq 1$.

details. First, the exact values of the optimum weights are not known analytically and, second, they will not predict the true function exactly. As a

result, the loss function has a lower (non-zero) bound, which takes the value 8.52×10^{-8} as can be seen from the graph in Figure 3.

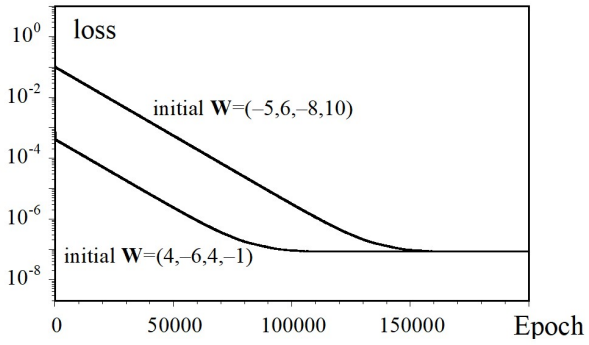

Fig. 3 The decrease of the loss function as a number of the epochs for the random initialization based on the weights $\mathbf{W} = (-5, 6, -8, 10)$ for a second learning process based on binomials weights $\mathbf{W} = (4, -6, 4, -1)$. The test function was chosen as $g(t) = 3\exp(-2t)\cos(70t) + \exp(2t^4)\cos(20t)$, which is not exactly predictable.

In Figure 3 we also compare again the learning process based on random and binomial initial weights. Once again, we can confirm that the efficiency of the learning process can be enhanced. In fact, after the same number of learning epochs, the loss associated with the better initial weights $\mathbf{W} = (4, -6, 4, -1)$ is almost three orders in magnitude lower.

In Figure 4 we compare the accuracy of the four numerically obtained weights, given by $\mathbf{W}(0.01) = (3.4630709, -4.9084044, 3.391008, -0.9639567)$ to predict the region $t \geq 0.5$ from 46 data obtained from $0 \leq t < 0.5$. We see that while for early times $0.5 \leq t \leq 0.75$ the agreement with the true values $g(t)$ is not so bad, for larger values the predictions are no longer so good and we see significant differences between the forecasted and the true values.

In closing, we should mention, if we had tested the scheme not based on an honest prediction, (where only for $t < 0.5$ the true values of g(t) were used) but for only a single-step prediction (where any past values were provided by the true values $g(t)$), then all of the "forecasted" data and the true graph $g(t)$ would be graphically indistinguishable from each other for this set of four weights, even though the function is not an epf.

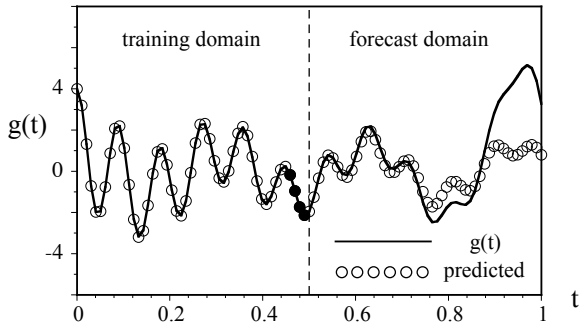

Fig. 4 Comparison of the forecasted data for $0.5 \leq t \leq 1$ based on optimized four weight factors and the true values of $g(t)$. The first 50 points for $0 \leq t < 0.5$ were used for the learning process leading to $\mathbf{W} = (3.4630709, -4.9084044, 3.391008, -0.9639567)$. The test function was chosen as $g(t) = 3\exp(-2t)\cos(70t) + \exp(2t^4)\cos(20t)$. It is not an exactly predictable function as the forecasted and exact data do not match.

3.2 Forecasting beyond a singularity

In this section we examine the usefulness of epfs to predict the value of functions that exhibit a singularity. Can we use the historical data on one side of the infinity for training such that the net can predict the future values of the function correctly on the other side of the singularity? We have two simple examples, both of the functions $f_1(t) = 3e^{2t}/(0.4875 - t)$ and $f_2(t) = 3e^{2t}/(0.4875 - t)^2$ have a singularity at $t = 0.4875$ and are positive in the training region $0 < t < 0.4785$.

We note, however, that neither function is an epf, such that a very large number N of inputs would be required to minimize the associated forecasting error. However, if we take the reciprocal of the input data $1/f(t - nh)$, then the underlying inverse data are epf, which means that the associated perfect weights would be fully sufficient to predict these data exactly.

We have demonstrated this in Figure 5a and 5b, where we used just two (three) historical data of the two functions $f_1(t)$ and $f_2(t)$ to predict their future values. The predictive accuracy is superb such that in both cases the singularity is not an obstacle for an accurate forecast.

The possibility to exactly forecast a time series even through a singularity has also direct applications for networks to be able of assigning finite values to the limit of diverging series. Since the early works of Borel [32], Hardy [33] and many others, it was demonstrated that the divergence of certain series

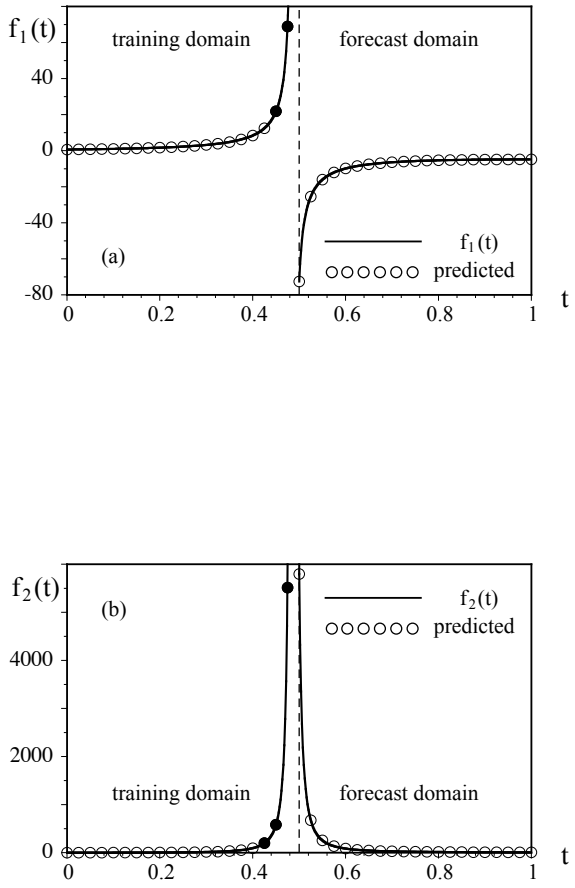

Fig. 5 (a) Comparison of the forecasted 20 values of a singular function for $0.5 \leq t \leq 1$ based solely on its two historical values at times $t = 0.450$ and 0.475 and the optimum weights $\mathbf{W}(0.025) = (1.902, -0.905)$ and the true values of $f_1(t)$. The future times are $t = 0.5 + (n-1)0.025$ with $n = 1, 2, ..., 20$. The test function was chosen as $f_1(t) = 3\exp(2t)/(0.4875 - t)$;(b) Same comparison, but the prediction is for a quadratic singularity, given by $f_2(t) = 3\exp(2t)/(0.4875 - t)^2$, requiring three historical data points at times $t = 0.425, 0.450$ and 0.475 and optimum weight factors given by $\mathbf{W}(0.025) = (2.854, -2.714, 0.8607)$.

does not automatically mean that it becomes useless. In fact, there have been several analytical and computational techniques developed that permit us to associate a finite value with a diverging series [34,35]. To provide a simple

example, the sequence $S \equiv \sum_{n=0}^{\infty}(-2)^n$ can be associated with the finite value $S = 1/3$. There are at least three different ways to show how this result can be understood. First, applying the distributive law $S \equiv 1 + 2(1 - 2 + 4 - 8 + 16 - ...) = 1 + 2S$ leads to the solution $S = 1/3$. Second, when we evaluate both sides of the geometric series $S = \sum_{n=0}^{\infty} t^n = 1/(1-t)$ for $t = -2$ we obtain $S = 1/3$. Third, there are also more general and systematic techniques such as the Borel summation [20], which is based on introducing a factorial factor $n! = \int_0^{\infty} dt\ \exp(-t)t^n$ into the summation $S = \sum_{n=0}^{\infty}(-2)^n n!/n!$ (in the numerator). If we replace the summation and integration, we obtain $S = \int_0^{\infty} dt\ \exp(-t)\sum_{n=1}^{\infty}(-2)^n t^n/n!$, whose (converging) summation portion can be evaluated as $\sum_{n=1}^{\infty}(-2)^n t^n/n! = \exp(-2t)$. As a result, we confirm again $S = \int_0^{\infty} dt\ \exp(-3t) = 1/3$.

We will illustrate that the forecasting power of neural nets can be exploited to evaluate diverging sums. The network is trained with the finite data obtained from the converging regime of the series. Once the weight factors are learned, they can be applied to the diverging region of interest.

Let us take as a concrete numerical example a sum that is not Borel summable such as $S = \sum_{n=0}^{\infty} 2^n$. We can compute the training set $S(t) = \sum_{n=0}^{\infty} t^n$ from the perfectly converged values for $t < 1$. While a more complicated example would require many training sets, the basic idea can be illustrated here for only two historical points $t_1 = 0$ and $t_2 = 0.5$ with $S_1 \equiv S(t_1) = 1$ and $S_2 \equiv S(t_2) = 2$. The associated optimum weights are $W = (2, -1)$. We assume $t_3 = 1, t_4 = 1.5$ and $t_5 = 2$ and want to find $S(t_5)$. For the first predicted point, we obtain $S_3^{-1} = W_1 S_2^{-1} + W_2 S_1^{-1}$ leading to $S_3^{-1} = 0$. A second iteration $S_4^{-1} = W_1 S_3^{-1} + W_2 S_2^{-1}$ leads to $S_4^{-1} = -1/2$. The final forecasting step $S_5^{-1} = W_1 S_4^{-1} + W_2 S_3^{-1}$ leads to $S_5^{-1} = -1$. As $S_5 = S(2) = \sum_{n=0}^{\infty} 2^n$, we have shown that the numerical value of this (non-Borel summable) diverging sum is equal to $\sum_{n=0}^{\infty} 2^n = -1$.

4 Generalization of epfs to multi-neuron networks

So far, our discussion was based entirely on using a single neuron with N input channels for forecasting. As we continuously increase the complexity of the network in a systematic way, one might expect that we can enlarge the possible class of epfs by increasing the number of neurons in the first layer.

For the case, where we do not allow for any activation function, one can easily see that due to the linear character of a network with M neurons, the class of those epfs discussed above is already complete, despite the much larger available space of weight factors W_{mn}, with $n = 1, 2, ..N$ and $m = 1, 2, ..M$. Here the predicted output value is given by $y_{out} = \sum_{m=1}^{M} V_m \sum_{n=1}^{N} W_{mn} f(t - nh)$, where the coefficients V_m, with $m = 1, 2, ..N$ are the weights associated with the output of each of the M neurons. As the same output y_{out} can be obtained equivalently via $y_{out} = \sum_{n=1}^{N} W_n^{eff} f(t - nh)$ any M-neuron system can be replaced equivalently by the single neuron system already studied (with effective weights $W_n^{eff} \equiv \sum_{m=1}^{M} V_m W_{mn}$). This proves that the apparent larger

number of degrees of freedom associated with a M-neuron system cannot be deployed to led to a larger class of epfs or more powerful forecasting algorithms for general functions $f(t)$.

The next level of complexity is achieved if we permit each neuron to have an activation function, such that its output modifies to $y_{out,m}{=}A[\sum_{n=1}^{N} W_{mn} f(t{-}nh)]$, such that the forecasted output of M neurons is given by $y_{out}{=}\sum_{m=1}^{M} V_m A[\sum_{n=1}^{N} W_{mn} f(t{-}nh)]$. Due to the non-linearity $A[a{+}b] \neq A[a]{+}A[b]$ inherent to any activation function, the question about the existence of epfs is much more complex for a general A.

However, for the special case where the activation function is a rectified linear unit, a 2-neuron system with N inputs channels has again the identical set of epfs as a single neuron without an activation function. A ReLU refers to the most commonly deployed activation function for the outputs of the convolutional neural network neurons [3] and is defined via the (non-differentiable) maximum function as $R[a] \equiv max[0, a]$. As it fulfills the identity $R[a] - R[-a] = a$, we see that a 2-neuron network with N inputs and an output given by $y_{out} = V_1 R[\sum_{n=1}^{N} W_{1n} f(t - nh)] + V_2 R[\sum_{n=1}^{N} W_{2n} f(t - nh)]$, can once again be mapped onto our original problem of a 1-neuron system (without ReLu) if $V_2 = -V_1$ and $W_{2n} = -W_{1n}$.

The interesting question therefore arise, if this 2-neuron system has a larger class epfs than the 1-neuron system. Are there more functions that can be exactly forecasted?

$$f(t) = V_1 R[W_{11} f(t{-}h){+}W_{12} f(t{-}2h)]{+}V_2 R[W_{21} f(t{-}h){+}W_{22} f(t{-}2h)] \quad (9)$$

In order to examine this question, we can attempt to reversely engineer the potential candidates by viewing Eq. (9) as a two-step iteration scheme, that for a given set of $\mathbf{W}$ and $\mathbf{V}$ iterates two initial values [corresponding to $f(t-2h)$ and $f(t-h)$] forward. The resulting sequence of iterated values for $f(t)$ and $f(t+nh)$, would then naturally be a discrete sampling of the underlying epf. In the absence of any activation function this scheme works well to generate any epf due to the inherent linearity of the system. However, due to the non-linearity and the non-invertibility of the ReLU, the sequence of iterated values $f(t{+}nh)$ for $n = 0, 1, 2, ...$ approaches certain periodic steady states. To have a concrete example, for the weight factors $\mathbf{W} = [(1, -1), (-1, 1)]$ and $\mathbf{V} = (1, 1)$, and initial values $[f(t{-}h), f(t{-}2h)] = (1, 3)$ or $(3, 1)$, the iteration approaches quickly a repeating three-point cycle $(1, 1, 0)$.

Even if the non-invertible ReLu activation function is replaced by an invertible logistic sigmoid function $L(z) \equiv 1/[1 + \exp(-z)]$, the resulting iterative scheme is very dissipative and approaches a one or two cycle steady state function that is rather independent of the choice of the initial values $f(t-h)$ and $f(t-2h)$. As a truly useful forecasting algorithm should not predict the identical future values $f(t+nh)$ for any different initial values, this analysis suggests that it is very nontrivial to generalize the concept of epfs to multi-neuron systems with activation functions. On the other hand, we also point

out that the low-weight limit of the logistic activation function approaches our original system, as $4/W_1[L(W_1 z) - 1/2] \sim z$ is linear. In this limit, the superposition principle for sums and products of epfs generalizes trivially to more complicated configurations of multiple neurons.

5 Summary and future challenges

We have shown that those subsets of functions $f(t)$, whose temporal derivatives can be partitioned into a finite number N of clusters containing linearly dependent functions, are *exactly predictable* by a neural net based on N inputs, associated with sampled values of the function at earlier times. Quite remarkable, it turns out that summation as well as products over these epfs are exactly predictable again. We have provided rules that permit us to compute the perfect weight factors for these new functions in terms of the weights of the original constituent functions.

Another interesting question concerns the possibility to generalize the existence of exactly predictable function to those that have more than just one argument t. For example, it might be quite interesting from a practical perspective to explore if there are also subset of functions $f(t, x)$ that can be forecasted exactly from sample values $f(t - nh, x - mh)$. In this case, are there also superposition laws?

In 2009, Schmidt and Lipson [36] have generalized symbolic regression schemes [37] of numerical data to find not only analytical expressions that describe the data, but to also construct the differential equations, whose solutions were represented by the sampled data. To find the underlying general laws (differential equations) is, of course, an important challenge in many areas of science. If a data set is found to be exactly predictable with N inputs, this suggests that the temporal derivatives can be related to each other and grouped into N clusters. This conclusion can also be viewed from the reverse perspective. If a data set samples a function that is a solution to a linear differential equation of order N, then it will be automatically an epf of class=N. This means that the functional space of data describing possible solutions to differential equations can be severely restricted. If a network with N inputs is able to learn perfect weights (which make the data exactly predictable), then one could conclude that the underlying function can be indeed a solution to a differential equation of order N. This information might be very beneficial for the above algorithms.

We have focused our attention in this work on simple feedforward networks with a single hidden layer. While the inclusion of a nonlinear activation function can be incorporated into our theoretical framework and exactly predictable function sets can be constructed here as well, we consider it an important goal for future work to explore, how the concept of exactly predictable function sets can be also helpful for other more complicated forecasting architectures such as neural networks with more than just one output, deeper nets with more than a single hidden layer or even recurrent nets. In contrast to the

ARIMA-type situations discussed here, these environments are intrinsically nonlinear due to multiple and nested activation functions and one likely has to rely on purely numerical means in discovering their properties of the epfs. These sets will likely be rather specific for each application, and it is not clear if the summation and product rules that we discussed above to construct more complicated function sets from simpler ones can be derived. As these deep machine learning environments have in general more predictive powers than the traditional ARIMA-based schemes, these explorations are worthwhile. These are especially true in view of the fact that so far these new powerful algorithms are basically still black box in nature.

Acknowledgements We appreciate Prof. N. Christensen's enthusiasm for this work at its early stages and numerous illuminating discussions. We also thank Prof. R.F. Martin and C. Gong for very helpful discussions and pointing out recent related work in the literature. This work has been supported by the US National Science Foundation and Research Corporation.

A : Proof of the superposition laws for sums

Here we briefly outline the basic ideas for the general proof of the superposition law that permits us to construct the perfect connection weights $W_k(F)$ for $F(t) \equiv f(t) + g(t)$ in terms of the original weights of $f(t)$, denoted by $W_n(f)$, and of $g(t)$, denoted by $W_m(f)$. We require

$$f(t) + g(t) = \sum_{k=1}^{N+M} W_k(f+g)[f(t-kh) + g(t-kh)] \tag{10}$$

$$f(t) = \sum_{n=1}^{N} W_n(f)\ f(t-nh) \tag{11}$$

$$g(t) = \sum_{m=1}^{M} W_m(g)\ g(t-mh) \tag{12}$$

The proof in full generality is very clumsy and can be best performed with some computer algebra software packages such as Mathematica or MatLab. The basic idea is to replace the $(M+1)$ functions at the times $f(t)$, $f(t-h)$ up to $f(t-Mh)$ in terms of the N functions at earlier times $f(t-jh)$ with $j = 1+M, N+M$. This iterative procedure that needs to be done in a strict consecutive order is extremely cumbersome. For example, by evaluating both sides of Eq. (11) for the argument $t - Mh$, we have

$$f(t-Mh) = \sum_{n=1}^{N} W_n(f)\ f(t-(M+n)h) \tag{13}$$

Similarly, for the argument $t-(M-1)h$ and insertion of Eq. (13) we obtain

$$f\ \ (t-(M-1)h) = \sum_{n=1}^{N} W_n(f)\ f(t-(M-1+n)h)$$

$$= W_1(f)\ f(t-Mh) + \sum_{n=2}^{N} W_n(f)\ f(t-(M-1+n)h)$$
$$= W_1(f) \sum_{n=1}^{N} W_n(f)\ f(t-(M+n)h) + \sum_{n=2}^{N} W_n(f)\ f(t-(M-1+n)h) \quad (14)$$

This sequence of iterative steps needs to be repeated $(M+1)$ times until the function $f(t)$ can be expressed in terms of $f(t-jh)$ with $j = M+1, N+M$ and all $W_n(f)$. The same replacements need to be performed for the function $g(t)$ as well. Here the $(N+1)$ functions at the times $g(t)$, $g(t-h)$ up to $g(t-Nh)$ need to be expressed in terms of the N functions at earlier times $g(t-jh)$ with $j = N+1, N+M$.

After these expressions are inserted into Eq. (10), this equation becomes finally a single linear equation for the unknown $(N+M)$ weights $W_k(f+g)$, containing all N weights $W_n(f)$ and M weights $W_m(f)$ as well as the N functions $f(t-jh)$ with $j = 1+M, N+M$ and M functions $g(t-jh)$ with $j = N+1, N+M$.

As this single equation needs to be satisfied for all times t, the $(N+M)$ pre-factors in front of all $f(t-jh)$ and $g(t-jh)$ need to vanish identically. The corresponding set of $(N+M)$ equations for the $(N+M)$ weights $W_k(f+g)$ can be solved uniquely.

To give the reader a better idea of the complexity of the derivation, we present here a concrete example, where we choose $N=3$ and $M=2$. For example, $f(t) = t^2\exp(3t)$ and $g(t) = \cos(5t)$ would fall in this category with the known weights according to Table 1.

$$f(t) + g(t) = \sum_{k=1}^{5} W_k[f(t-kh) + g(t-kh)] \quad (15)$$
$$f(t) = U_1\ f(t-h) + U_2 f(t-2h) + U_3\ f(t-3h) \quad (16)$$
$$g(t) = V_1\ g(t-h) + V_2\ g(t-2h) \quad (17)$$

where for notational simplicity we abbreviate $U_n \equiv W_n(f)$ and $V_m \equiv W_m(g)$. Using Eqs. (16) and (17) repeatedly, the sequence of required replacements leads to

$$\begin{aligned}
f(t) &= [U_1^3 + 2U_1U_2 + U_3]f(t-3h) + [U_2(U_1^2+U_2) + U_1U_3]f(t-4h) \\
&\quad + (U_1^2+U_2)U_3 f(t-5h) \\
f(t-h) &= [U_1^2+U_2]f(t-3h) + (U_1U_2+U_3)f(t-4h) + U_1U_3 f(t-5h) \\
f(t-2h) &= U_1 f(t-3h) + U_2 f(t-4h) + U_3 f(t-5h) \\
g(t) &= (V_1^4 + 3V_1^2V_2 + V_2^2)\ g(t-4h) + V_1V_2(V_1^2+2V_2)\ g(t-5h) \\
g(t-h) &= (V_1^3V_2 + 2V_1V_2)\ g(t-4h) + V_2(V_1^2+V_2)\ g(t-5h) \\
g(t-2h) &= (V_1^2+V_2)\ g(t-4h) + V_1V_2\ g(t-5h) \\
g(t-3h) &= V_1\ g(t-4h) + V_2\ g(t-5h) \quad (18)
\end{aligned}$$

As a result, we obtain for Eq. (15)

$$\begin{aligned}
0 &= -f(t) - g(t) + \sum_{k=1}^{5} W_k[f(t-kh) + g(t-kh] \\
&= A_1 f(t-3h) + A_2 f(t-4h) + A_3 f(t-5h) + A_4\ g(t-4h) + A_5\ g(t-5h) \quad (19)
\end{aligned}$$

where the five coefficients are given by

$$A_1 = -U_1^3 - U_3 - 2U_1U_2 + (U_1^2+U_2)W_1 + U_1W_2 + W_3$$

$$
\begin{aligned}
A_2 &= -U_2^2 - U_1(U_1U_2 + U_3) + (U_1U_2 + U_3)W_1 + U_2W_2 + W_4 \\
A_3 &= -U_2U_3 - U_1U_1U_3 + U_3U_1W_1 + U_3W_2 + W_5 \\
A_4 &= -V_1^4 - 3V_1^2V_2 - V_2^2 + (V_1^3 + 2V_1V_2)W_1 + (V_1^2 + V_2)W_2 + V_1W_3 + W_4 \\
A_5 &= -V_1^3V_2 - 2V_1V_2^2 + (V_1^2V_2 + V_2^2)W_1 + V_1V_2W_2 + V_2W_3 + W_5
\end{aligned} \tag{20}
$$

If we equate these five coefficients A_k to zero, we obtain the final solutions for the weights W as

$$
\begin{aligned}
W_1 &= U_1 + V_1 \\
W_2 &= U_2 + V_2 - U_1V_1 \\
W_3 &= U_3 - U_1V_2 - U_2V_1 \\
W_4 &= -U_2V_2 - U_3V_1 \\
W_5 &= -U_3V_2
\end{aligned} \tag{21}
$$

In view of the complexity of the expressions in the intermediate steps, these forms are remarkably simple and they are in full agreement with the general solutions of Eq. (7) for arbitrary N and M.

B : Weight factors for the product rule

Here we briefly outline the basic ideas for the general proof of the superposition law that permits us to construct the perfect connection weights $W_k(F)$ for products $F(t) \equiv f(t)g(t)$ in terms of the original weights of $f(t)$, denoted by $W_n(f)$, and of $g(t)$, denoted by $W_m(f)$. We require

$$
f(t)\ g(t) = \sum_{k=1}^{NM} W_k(fg) f(t - kh)\ g(t - kh) \tag{22}
$$

$$
f(t) = \sum_{n=1}^{N} W_n(f)\ f(t - nh) \tag{23}
$$

$$
g(t) = \sum_{m=1}^{M} W_m(g)\ f(t - mh) \tag{24}
$$

The approach to derive of the weights $W_k(fg)$ from the $W_k(f)$ and $W_k(g)$ is -in principle- similar to the one used in appendix A, but it is significantly more complicated and we illustrate it here only for the $N = 3$ and $M = 3$ case. Here we would iteratively use Eq. (23) to replace the functions f and g at later times in terms of the six values $f(t-kh)$ and $g(t-kh)$ for $k = 7, 8$ and 9. After the replacements, the central equation Eq. (23) for the nine weights $W_k(fg)$ depend on the nine time-dependent product functions $f(t - k_1h)g(t - k_2h)$ with $k_1 = 7, 8, 9$ and $k_2 = 7, 8, 9$. If we assume that these nine functions are linearly independent of each other, we have to require that the corresponding nine pre-factors vanish. If we solve the resulting nine coupled but linear equations of the nine weights $W_k(fg)$ for $k = 1, 2, ..., 9$ we finally obtain the solutions

$$
W_1(fg) = U_1V_1 \tag{25}
$$

$$
W_2(fg) = U_1^2V_2 + U_2V_1^2 + 2U_2V_2 \tag{26}
$$

$$
W_3(fg) = U_3(V_1^3 + 3V_1V_2 + 3V_3) + U_1(U_2V_1V_2 + U_1^2V_3 + 3U_2V_3) \tag{27}
$$

$$
W_4(fg) = U_1^2U_2V_1V_3 - U_2^2(V_2^2 - 2V_1V_3) + U_1U_3[V_2(V_1^2 + 2V_2) + V_1V_3] \tag{28}
$$

$$W_5(fg) = -U_1U_2^2V_2V_3 + U_1^2U_3(V_1^2 + 2V_2)V_3 + U_2U_3(-V_1V_2^2 + 2V_1^2V_3 - V_2V_3) \quad (29)$$
$$W_6(fg) = U_2^3V_3^2 - U_1U_2U_3V_3(V_1V_2 + 3V_3) + U_3^2(V_2^3 - 3V_1V_2V_3 - 3V_3^2) \quad (30)$$
$$W_7(fg) = U_3V_3[U_2^2V_1V_3 + U_1U_3(V_2^2 - 2V_1V_3)] \quad (31)$$
$$W_8(fg) = -U_2U_3^2V_2V_3^2 \quad (32)$$
$$W_9(fg) = U_3^3V_3^3 \quad (33)$$

For notational simplicity, we have used again the abbreviations $U_k \equiv W_k(f)$ and $V_k \equiv W_k(g)$. Unfortunately, we have not been able to recognize a certain regular pattern to these nine weights that would have allowed us to predict the corresponding 16 weights for the $N = 4$ $M = 4$ system. Even though we note that the sum of the indices of each factor U and V matches the index k of $W_k(fg)$, respectively, to predict reliably the corresponding permutations of these factors, their pre-factors and signs seems difficult.

C : Optimum weights involving a sum of products

Here we examine the optimum weights due for the specific function $f(t)$

$$f(t) = 3\ \exp(a_1t)\ \cos(b_1t) + \exp(a_2t)\cos(b_2t) \quad (34)$$

where we have used the specific values $a_1 = -2$ and $a_2 = 2$ for the decay and growth rates and $b_1 = 70$ and $b_2 = 20$ for the two frequencies. As a sum of the class=2 functions $\exp(a_1t)\cos(b_1t)$ and $\exp(-a_2t)\cos(b_2t)$, the function $f(t)$ is again a epf of class=4. Applying consecutively the superposition laws derived in this work for optimal weights for the summation and products of functions (Eqs. 7, 25 and 26), one can derive the following analytical expressions for the six optimal weights.

$$W_1 = 2\exp(a_1h)\cos(b_1h) + 2\exp(a_2h)\cos(b_2h) \quad (35)$$
$$W_2 = -\exp(2a_1h) - \exp(2a_2h) - 4\exp(a_1h + a_2h)\cos(b_1h)\cos(b_2h) \quad (36)$$
$$W_3 = 2\exp(a_1h + 2a_2h)\cos(b_1h) + 2\exp(2a_1h + a_2h)\cos(b_2h) \quad (37)$$
$$W_4 = -\exp(2a_1h + 2a_2h) \quad (38)$$

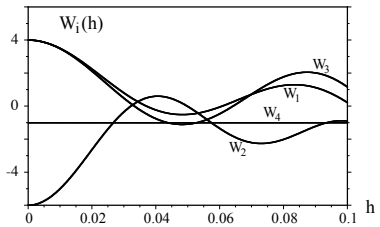

Fig. 6 The dependence of the optimal four weights for the test function Eq. (34) for $a_1 = -2$, $a_2 = 2$, $b_1 = 70$ and $b_2 = 20$ as a function of the grid spacing h.

In Figure 6 we have graphed these four optimum weights as a function of the grid spacing h. In the limit of small spacings $h \to 0$ we find that the weights approach the values $(W_1, W_2, W_3, W_4) = (4, -6, 4, -1) \equiv \mathbf{W}(h = 0)$. This set corresponds precisely the optimal (h-independent) weights B_{n4} for any polynomial of degree 3. This is not a coincidence as the original optimal weights of the constituent functions $\exp(at)$, $\cos(bt)$ of $f(t)$ of Eq. (34) converge already in this limit to the alternating binomial coefficients given in Eq. (8).

The key question is, whether the binomial coefficients can act as helpful initial values for the learning algorithm for the relevant case where $h \neq 0$. For example, for $h < 0.0076$, each the optimal set of weights differs from the set $\mathbf{W}(h = 0)$ by at most 10%. For example, for $h = 0.01$ we have the set of exact optimal weights given by $\mathbf{W}(0.01) = (3.50, -5.00, 3.48, -1.00)$, very similar to $B_{n4} = (-1)^{n+1}(4, n)$. This suggests, that as long as the grid spacing is not too large, the binomial set should be an ideal set of weight parameters in order to initialize the net for the learning process.

References

1. Weigend AS, Gershenfeld NA. Time Series Prediction: Forecasting the Future and Understanding the Past. Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley: 1993.
2. Hill T, O'Connor M, Remus W. Neural Network Models for Time Series Forecasts. Management Science 1996;42(7):1082-1092.
3. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press, 2016.
4. James W. The Principles of Psychology, Henry Holt and Company,1890.
5. McCulloch W, Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 1943;5:115-133.
6. Hebb D. The Organization of Behavior. Wiley, New York,1949.
7. Rosenblatt F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review 1958;65(6):386-408.
8. Minsky M, Papert S. Perceptrons. MIT Press, Cambridge MA, 2nd edition 1972 first edition 1969.
9. Rummelhart DE, Hinton GE, Williams RJ. Learning Representations by Back-Propagating Errors. Nature 1986;323:533-536.
10. Rojas R. The Backpropagation Algorithm Neural Networks: A Systematic Introduction. Berlin: Springer, 1996.
11. Mills TC. Time Series Techniques for Economists. Cambridge University Press, 1990.
12. Percival DB, Walden AT. Spectral Analysis for Physical Applications. Cambridge University Press, 1993.
13. Hamilton J. Time Series Analysis. Princeton University Press, 1994.
14. Papoulis A. Probability, Random Variables, and Stochastic Processes. Tata McGraw-Hill Education, 2002.
15. Box GEP. Time Series Analysis: Forecasting and Control. Wiley, 2015.
16. Hamilton JD. Time Series Analysis. Princeton, NJ: Princeton University Press, 1994.
17. Hornik K, Stinchcombe M, White H. Multilayer Feedforward Networks are Universal Approximators. Neural Networks. 1989;2:359-366.
18. Hornik K, Stinchcombe M, White H. Neural networks. Universal Approximation of an Unknown Mapping and Its Derivatives using Multilayer Feedforward Networks. 1990;3(5): 551-560.
19. Leshno M, Lin VY, Pinkus A, Schocken S. Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate any Function. Neural Networks. 1993;6(6): 861-867.
20. Barron AE. Universal Approximation Bounds for Superpositions of a Sigmoidal Function. IEEE Trans. on Information Theory. 1993;39:930-945.
21. For a nice visual proof of the universality theorem, see chapter 4 in Nielson M. Neural Networks and Deep Learning. http://neuralnetworksanddeeplearning.com/
22. Oppenheim AV, Schafer RW. Discrete-Time Signal Processing. Prentice Hall, Englewood Cliffs NJ, 1989.

23. Kamen EW. Introduction to Signals and Systems. McMillan, NY, 1990.
24. Gershenfeld NA. The Nature of Mathematical Modeling. Cambridge Press, Cambridge, 1999.
25. Glorot X, Bengio Y, see page 249 in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), Italy. 2010;9 of JMLR: W&CP 9.
26. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2014;arXiv:1412.6980v9.
27. Zhang Z, Ma L, Li Z, Wu C. Normalized Direction-preserving Adam. 2017;arXiv:1709.04546v2.
28. Reddi SJ, Kale S, Kumar S. On the Convergence of Adam and Beyond. 2018;arXiv:1904.09237.
29. Loshchilov I, Hutter F. Fixing Weight Decay Regularization in Adam. 2017;arXiv:1711.05101v2.
30. N.S. Keskar and R.Socher. Improving Generalization Performance by Switching from Adam to SGD. arXiv:1712.07628v1 (2017).
31. We used the Keras package in python with tensor flow background as well the neural network environment provided by Mathematica and obtained consistent data.
32. Borel E. Memoire sur les Series Divergentes. Ann. Sci. Ec. Norm. Super., Series 1899;3(16): 9-131.
33. Hardy GH. Divergent Series. AMS Chelsea, New York 1949.
34. Lisowski C, Norris S, Pelphrey R, Stefanovich E, Su Q, Grobe R. Ground State Energies from Converging and Diverging Power Series Expansions. Ann. Phys. 2016;373:456-469.
35. Lv QZ, Norris S, Pelphrey R, Su Q, Grobe R. Computation of Diverging Sums Based on a Finite Number of Terms. Comp. Phys. Comm. 2017;219:1-12.
36. Schmidt M, Lipson H. Distilling Free-Form Natural Laws from Experimental Data. Science 2009;324, 81-85.
37. Bongard J, Lipson H. Automated Reverse Engineering of Nonlinear Dynamical Systems. Proc. Natl. Acad. Sci. U.S.A. 2007;104(24):9943-9948.