Optimal Neural Network Feature Selection for Spatial-Temporal Forecasting

E. Covas[1, a)] and E. Benetos[2, b)]
[1)] *CITEUC, Geophysical and Astronomical Observatory, University of Coimbra, 3040-004, Coimbra, Portugal*
[2)] *School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, London E1 4NS, U.K.*

(Dated: 10 May 2019)

Neural networks, and in general machine learning techniques, have been widely employed in forecasting time series and more recently in predicting spatial-temporal signals. All of these approaches involve some kind of feature selection regarding what past data and what neighbour data to use for forecasting. In this article, we show extensive empirical evidence on how to independently construct the optimal feature selection or input representation used by the input layer of a feed forward neural network for the purpose of forecasting spatial-temporal signals. The approach is based on results from dynamical systems theory, namely non-linear embedding theorems. We demonstrate it for a variety of spatial-temporal signals, and show that the optimal input layer representation consists of a grid, with spatial/temporal lags determined by the minimum of the mutual information of the spatial/temporal signals and the number of points taken in space/time decided by the embedding dimension of the signal. We present evidence of this proposal by running a Monte Carlo simulation of several combinations of input layer feature designs and show that the one predicted by the non-linear embedding theorems seems to be optimal or close to being optimal. In total we show evidence in four unrelated systems: a series of coupled Hénon maps; a series of coupled ordinary differential Equations (Lorenz-96) phenomenologically modelling atmospheric dynamics; the Kuramoto-Sivashinsky equation, a partial differential equation used in studies of instabilities in laminar flame fronts and finally real physical data from sunspot areas in the Sun (in latitude and time) from 1874 to 2015. These four examples cover the range from simple toy models to complex non-linear dynamical simulations, and real data. Finally, we also compare our proposal against alternative feature selection methods, and show that it also works for other machine learning forecasting models.

Machine learning techniques, and in particular neural networks, have been used to forecast spatial-temporal signals. However, there is a need to calibrate many parameters and perform a feature selection. This article proposes that non-linear dynamical systems theory provides the methods to establish *a priori* what is the feature selection of past data and neighbour data to use for optimal forecasting.

I. INTRODUCTION

Given a physical data set, one of the most important questions one can pose is: "Can we predict the future?" This question can be put forward irrespectively of the fact that we may already have some insight or even be certain on what the exact model behind some or all the observed variables is. For example, for chaotic dynamical systems[1,2], we may even have the underlying dynamics but still find it hard to predict the future, given that chaotic systems have exponential sensitivity to initial conditions. The more chaotic a system is (as measured by the positiveness of their largest Lyapunov exponents[3,4]) the harder it gets to predict the future, even within very short time horizons. In the limit case of a random system, it is not possible to predict the future at all, although one can opine on certain future statistics[5]. For the case of weakly chaotic systems, there is extensive literature on forecasting methods ranging from linear approximations[6]; truncated functional expansion series[7,8]; non-linear embeddings[9]; auto-regression methods[10]; hidden Markov models[11] to state-of-the-art neural networks and deep learning methodologies[12] and many others, too long to list here.

Most literature on forecasting chaotic signals is dedicated to a single time series, or by treating a collection of related time series as a non-extended set, i.e. a multivariate set of discrete variables as opposed to a spatially continuous series. For forecasting spatial-temporal chaos we refer the reader to[13–23] and references therein. Even rarer are attempts to forecast spatial-temporal chaos using neural networks and deep learning methodologies[24–34], although this field of research is clearly growing at the moment[35]. Nonetheless, this area of research is of importance, as most physical systems are spatially extended, e.g. the atmospheric system driving the Earth's weather[36]; the solar dynamo driving the Sun's sunspots[37]; and the influence of sunspots on the Earth's magnetic field via the solar wind, coronal mass ejections and solar flares – the so-called space weather[38], which may have real economic implications[39]. Nonetheless its importance, forecasting spatial-temporal chaos is difficult. The reasons are many, but mainly: first, the geometric dimension of the attractor[40] – usually quite large, the so-called curse of dimensionality[41]; and second how to choose the variables to use for forecasting, i.e., is there enough information on the same point back in time to

[a)] Also at School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, London E1 4NS, U.K.; Electronic mail: eurico.covas@mail.com.; http://www.euricocovas.com.
[b)] Electronic mail: emmanouil.benetos@qmul.ac.uk.

derive the future of that particular point, or do spatial correlations and spatial propagation affect it in a way that one must take into account some spatial and temporal neighbours set to forecast the future. If this is the case, can that set of points be defined and how can it be constructed? It is this last question that we investigate in this article, in the particular context of forecasting using neural networks.

Feature extraction and the design of the input representation of the input layer of a neural network is considered to be an art form, relying mostly on trial and error and domain knowledge (see[42] for examples and references). For time series forecasting, the simplest approach consists of designing the input layer as a vector of previous data using a time delay, the so called time delay neural network method[43–49]. For spatial-temporal series, one can generalize it to include temporal and spatial delays[22,24]. This is where the connection to dynamical systems can be useful.

In 1981, Takens established the theoretical background[50] in his embedding theorem for a mathematical method to reconstruct the dynamics of the underlying attractor of a chaotic dynamical system from a time ordered sequence of data observations. Notice the reconstruction conserves the properties of the original dynamical system up to a diffeomorphism. Further developments established a series of theorems[51–53] that provided the basis for a non-linear embedding and forecasting on the original variables. The embedding theorem has now been extended to the continuous variable case, i.e., a spatial-temporal signal[54,55]. The theorems and related articles propose to use a time delay approach with the time lag based on the first minima of the *mutual information*[56] – see[57–59] – and to choose the number of points to include using the method of *false nearest neighbours* detection suggested by[60] and reported in detail in[58,61–63].

Some authors discuss the use of either the mutual information and/or embedding dimension as a constraint on feature representation (see e.g.[64]). There are also authors[65] that try to use neural networks to determine the optimal embedding and time delay for the purpose of local reconstruction of states with a view to forecast (the opposite of what we try to empirically demonstrate here). Fig. 6 in[66] shows how the forecasting error for a pure time series prediction changes with the delay and the number of time delay points used as an input – they use a reinforcement learning based dimension and delay estimator to derive the best dimension and delay, but do not seem to show that it is the dynamical systems' derived values that are indeed optimal for forecasting, neither do they show any extension to spatial-temporal signals as we demonstrate in this article. Finally, other authors[20] try to use support vector machines (SVMs) to forecast spatial-temporal signals and use delays and embedding approaches to define the state vectors. In fact, Parlitz and Merkwirth mention in their article[17] that local reconstruction of states "... may also serve as a starting point for deriving local mathematical models in terms of polynomials, radial basis functions or neural networks...". Here we attempt to show empirical evidence that this is not just a starting point, but the optimal neural network input feature selection.

To the authors' knowledge, all the references on neural network forecasting of spatial-temporal dynamics that use the embedding theorems and the related mutual information and the false nearest neighbours methods seem not to justify its use, i.e., the approach is explained, even suggested to be optimal, but neither proven theoretically or empirically. In this article, we attempt to provide an empirical evidence for this optimality. Using this theoretical framework, we propose that this non-linear embedding method, using the training data alone without reference to the forecasting model, can be used to indicate the best way to construct the feature representation for the input layer of a neural network used for forecasting both in space and time. In order to support this proposal, we, in this article, show empirical evidence for an optimal feature selection for four particular cases of two-dimensional spatial-temporal data series s_m^n, where by two-dimensional we mean a scalar field that can be defined by a $N \times M$ matrix with components $s_m^n \in \mathbb{R}$. The goal of this article is not to demonstrate the ability to forecast, which has already been done by several authors in the literature above-mentioned, but rather that there is no need to calibrate the neural network feature selection specification by the "dark art" of trial and error or any other alternative method.

The article is divided as follows. In section II we explain our forecasting model, in section III we describe our proposal, in section IV we show our results supporting it, in section V A we explore alternatives such as other feature selection approaches and deeper neural network architectures and compare them against our proposal, and finally in section VI we make our concluding remarks.

II. MODEL

The neural network architecture we chose to demonstrate our proposal is a form of the basic feed-forward neural network, sometimes called the time-delayed neural network[43], trained using the so-called back-propagation algorithm[67–70]. We coded our own neural network model and back-propagation without relying on existing neural network libraries and all results in this article are based on that model unless otherwise explicitly mentioned. We focus on spatial-temporal series, so we have extended the usual time-delayed neural network to be a time and space delayed network. The overall feature representation of the network is depicted in detail in Fig. 1. Notice we chose to use feed-forward neural networks rather than more complex neural networks such as recurrent neural networks[71], since feed-forward ones are simpler to design; are capable of being used for forecasting of even complex chaotic signals; are guaranteed to converge, at least, to a local minima; and are easier to interpret.

Under this input representation, we use the ideas proposed in[17,18] to construct a grid of input values which are then fed to the neural network to produce a single output, the future state. Formally, we start with $n = 1,...,N$ and $m = 1,...,M$. Given a spatial-temporal series $\mathbf{s}$ which can be defined by a $N \times M$ matrix with components $s_m^n \in \mathbb{R}$. These components we will call *states* of the spatial-temporal series. Given a number $2I \in \mathbb{N}$ of neighbours in space of a given s_m^n and a number $J \in \mathbb{N}$ of

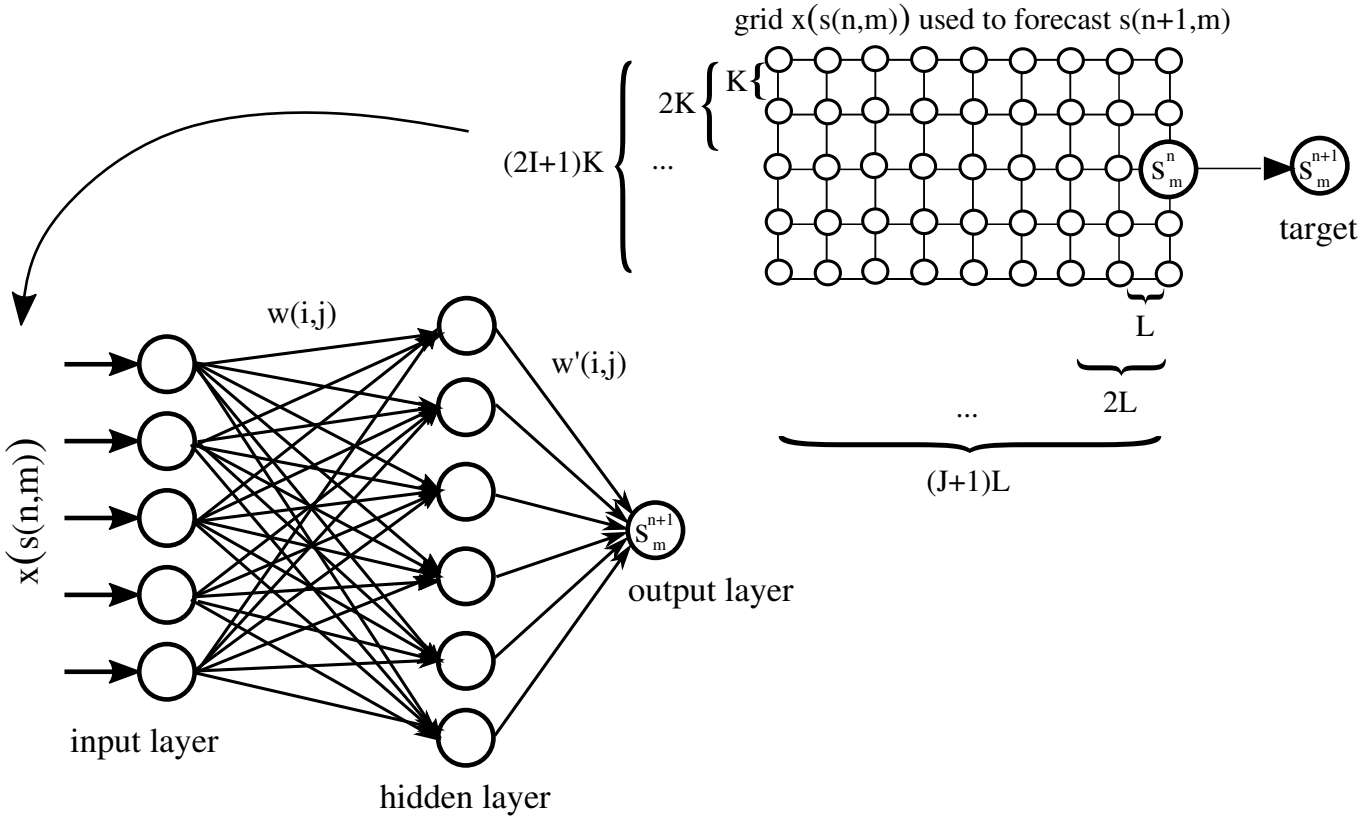


FIG. 1: Neural network architecture for forecasting spatial-temporal signals. The neural network is made of an input layer, one or more hidden layer(s) and one output layer. In this article, for simplicity, we use only one hidden layer and the output layer is made of a single neuron. Each input pattern $x(i)$ is sent to the input layer, then each of the hidden neurons' values is calculated from the sum of the product of the weights by the inputs $\sum w(i,j)x(i)$ and passed via the non-linear activation function. Then the output is calculated by the product of the second set of weights times the hidden node values $\sum w'(i,j)y(i)$ again passed to another (or the same) activation function. Each input pattern $x(i)$ is actually a matrix constructed using an embedding space of spatial and temporal delays, calculated from the actual physical spatial-temporal data values $s(n,m)$. After many randomly chosen input patterns are passed via the neural network, the weights hopefully converge to an optimal training value. One can then forecast using the last time slice of the training set, and compare against the test set, the real future data set. Then the forecasted data is concatenated and the process repeated to forecast more than one time slice.

temporal past neighbours relative to s_m^n (see Fig. 1 for details). For each s_m^n, we define the input (feature) vector $\mathbf{x}(s_m^n)$ with its components given by s_m^n, its $2I$ neighbours in space and its J neighbours in time, and with K and L representing the spatial and temporal lags:

$$\begin{aligned}\mathbf{x}(s_m^n) = \quad & \{s_{m-IK}^n,\ldots,s_m^n,\ldots,s_{m+IK}^n, \\ & s_{m-IK}^{n-L},\ldots,s_m^{n-L},\ldots,s_{m+IK}^{n-L},\ldots \\ & \ldots,s_{m-IK}^{n-JL},\ldots,s_m^{n-JL},\ldots,s_{m+IK}^{n-JL}\}.\end{aligned} \tag{1}$$

So, the input $\mathbf{x}(s_m^n)$ is a vector of dimension $(2I+1)(J+1)$ and to train the network we consider the target (output) to be the value s_m^{n+1}. We train the network using stochastic gradient back-propagation by running a stochastic batch where we randomly sample pairs of inputs and outputs from the training set: $\mathbf{x}(s_m^n)$ and s_m^{n+1}, respectively. Then at test time we choose inputs $\mathbf{x}(s_m^n)$, such that $n = N_{train}$, with N_{train} being the number of temporal slices on the training set. As for the remaining architecture, we use one hidden layer with N_h nodes. Regarding the back-propagation hyper-parameters, we included an adaptive learning rate $\eta_n = \eta/(1+n/10000)$, where the hyper-parameter η is the initial learning rate and η_n is the learning rate used at time step n. We included a momentum term α for faster convergence. A further hyper-parameter is the choice of the activation function (see[72]), we use either a ReLU (rectified linear unit) or a logistic sigmoid function depending on the dataset example we are working with. We also normalize the data before passing it through the neural network, in most cases we scale it in linear fashion $x \to \alpha_{nor} + x/\beta_{nor}$, and in the case of real physical data as we will see later, we scale it in logarithmic fashion it by $x \to \alpha_{nor} + \ln(1+x)/\beta_{nor}$, where x is the initial data, and α_{nor} and β_{nor} are the arbitrary shift and scaling constants, respectively. For the weight (and bias) initialization we choose random numbers with a constant distribution between $[0,1]$ and shifted by α_{rng} and scaled by β_{rng}. The final hyper-parameter is the number of epochs taken on the stochastic gradient descent which we denote by N_{steps}. All of these hyper-parameters are calibrated and fixed before we

do any simulations with respect to the parameters I, J, K, L, which are auto-calibrated by the above mentioned methods derived from dynamical systems theory. In this sense, I, J, K, L are not hyper-parameters of the neural network. We use the standard loss function $\mathscr{L} = \left(s_m^{n+1} - \hat{s}_m^{n+1}\right)^2$ for a prediction $\hat{s}_m^{n+1}$ centred around s_m^n against the real future value s_m^{n+1}.

III. PROPOSAL

Once we do a forecast, we then compare the goodness of fit by first visual inspection and second by numerically calculating the so-called structural similarity index SSIM(x,y) which has been proposed by[73] and used already in the context of spatial-temporal forecasting in[22,2474]. The SSIM index allows two images to be compared and provides a value of their similarity - a value of SSIM $= 1$ corresponds to the case of two perfectly identical images. We use it by calculating the SSIM(x,y) between the entire test set and the forecast set, since these can be interpreted as images (one spatial dimension/one temporal dimension).

Here we propose that the optimal time delay/spatial delays (L and K, respectively) must be the ones based on the first minima of the mutual information[57–59] and that the optimal number of temporal/spatial points to use (J and I, respectively) must be the ones based on the method of false nearest neighbours detection[58,60–63]. The mutual information is calculated by taking a s^i, a one-dimensional data set, and s^{i+L}, the related L-lagged data set. Given a measurement data point s^i, the amount of information $I(L)$ is given by the number of bits on s^{i+L} that can be predicted on average. We then take the mean of the mutual information over space and call it $\langle I(L)\rangle$, which is independent of K due to the averaging. After that, we take the first minimum of $\langle I(L)\rangle$, or, in the absense of a clear minimum, take the L temporal lag for which the $\langle I(L)\rangle$ drops significantly and starts to plateau. This gives us L^*, the optimal time delay. Conversely, we obtain K^* by calculating the spatial lag K for which we obtain the first minima of the time-averaged mutual information $\langle I(K)\rangle$. Once the optimal spatial and temporal lags K^* and L^* are calculated, we calibrate the minimum embedding dimension, or in other words, the number of space-time neighbours in the optimal phase space reconstruction. We employ the method of false nearest neighbours[60–62], which determines that falsely apparent close neighbours have been eliminated by virtue of projecting the full orbit in a increasing higher dimensional embedding phase space. **In practice, one has to set an arbitrary threshold for when the fraction of false nearest neighbours becomes "close" to zero, and this may not work for data with high noise-to-signal ratios. Therefore, we only consider, in this article, cases where the noise element is considered to be small (or in the case of synthetic signals, negligible).**

This gives us the J^*, the optimal number of time slices to take, and I^*, the optimal number of spatial slices to take in our $\mathbf{x}(s_m^n)$ reconstruction.

Technically, the results in these references, for the mutual information and the false nearest neighbours methods, giving an optimal value for the delay(s) and the number of delays, do not have a rigorous mathematical proof behind them, but are known to work well in practice. As we shall see later in the results section, small deviations from the optimal values given by these methods have a small impact in the predictability of the forecast.

In this article, we propose that as any set of input representation "approaches" the optimal one, then SSIM $\to 1$. In the case of finite training sets and/or noisy training sets SSIM $\to x < 1$, where x is the best forecast possible given the data set. Visually, we believe that the SSIM versus some reasonable metric constructed to represent the distance between all other input representations and the optimal input representation will show a skewed bell shape as depicted in Fig. 2, **i.e. small changes in the delay or number of delay values have a small impact on the predictability of the method, and conversely large changes have a large detrimental impact on the predictability**. In this proposal, we use the most obvious candidate to represent the distance between any input representation and the optimal input representation, the **scaled** Euclidian distance given by

$$d_e = \sqrt{\left(\frac{I-I^*}{I+I^*}\right)^2 + \left(\frac{J-J^*}{J+J^*}\right)^2 + \left(\frac{K-K^*}{K+K^*}\right)^2 + \left(\frac{L-L^*}{L+L^*}\right)^2}, \tag{2}$$

where I, J, K, L are the parameters for each representation and I^*, J^*, K^*, L^* are the ones derived from the dynamical systems theory. We also verified that other reasonable metrics, in particular the Manhattan distance[75], did not change the results qualitatively. We believe that using the Euclidean (and Manhattan) distance, which assumes both spatial and temporal dimensions are equally important, is a reasonable assumption, given that the datasets studied here are intrinsically spatial-temporal series with complex temporal and spatial non-linear interactions.

IV. RESULTS

In order to empirically substantiate our proposal, we take four examples of spatial-temporal series and attempt to forecast using our feed-forward neural network. First, we split the data into a training and a test set. Second, using the training set only, we calculate the optimal time delay/spatial delays (L^* and K^*, respectively) using the first minima of the mutual information, and then we calculate the optimal number of temporal/spatial points to use (J^* and I^*, respectively) using the method of false nearest neighbours. Only then we build the neural network model, calibrating the hyper-parameters of the network by exhaustive search on the parameter space to minimize the error on the training set. Then having fixed those hyper-parameters, we run a Monte Carlo simulation, retraining the network and running it to predict a forecast set, on each one of our four examples, sampling random values of the key feature selection parameters: I, J, K, L (including the trivial ones with $I = 0$ and/or $J = 0$) and calculate the values $d_e(I,J,K,L)$ and SSIM(I,J,K,L). We plot the latter as a function of the former to compare against our proposal as depicted

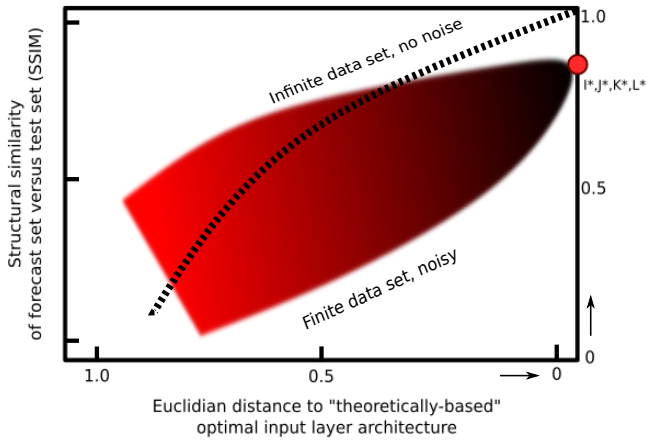


FIG. 2: Our main proposal. For a infinite noiseless training set, the SSIM approaches SSIM $\rightarrow 1$. For real data sets, there is a dispersion of the SSIM versus some reasonable metric constructed to represent the distance between any feature selection (e.g. d_e **as in Eq. 2**) and the optimal one given by dynamical systems theory. **Small (large) deviations in the delays and/or number of delays, represented by a small (large) distance d_e, produce a small (large) deviation in predictability, away from the optimal one**.

in Fig. 2.

We first take a physical system example, a real data example, and then we progress from "simpler" systems (coupled maps) capable of generating spatial-temporal chaos to more "complex" systems (coupled Ordinary Differential Equations - ODEs) to "really complex" systems (Partial Differential Equations - PDEs). This is partially motivated by results in the literature that show that general universalities are present in different levels of simplification of physical models[76], from the original PDEs to truncated ODE expansions (e.g. spectral method expansions[77]) to the most extreme simplification or discretization such as maps which capture the essence of the problem(s). In all cases we take examples with one spatial and one temporal dimension. However, we believe that our proposal will extend to multiple spatial dimensions. Again, notice that here we are not trying to demonstrate that neural networks, and in particular feed-forward neural networks can perform well in predicting spatial-temporal chaos (as this has been demonstrated in the literature already), but rather to show that the optimal choice of the input layer features is given by dynamical systems theory and does not need to be another neural network hyper-parameter calculated by the "dark art" of trial and error or any other alternative method.

A. Sunspot data - a physical system example

The first example we take is a physical real data example based on a previous article of one of us[24], where a neural network using the type of input representation above (Fig. 1) was used to forecast sunspot areas $A(t,\theta)$ in our Sun in both space (θ latitude) and time (Carrington Rotation index[78]). This sunspot data is usually called the "butterfly diagram" due to its butterfly wings like appearance[79]. One can see how this butterfly diagram looks like in[80].

Sunspot data is regularly seen as a benchmark for time series forecasting, given its chaotic nature and that it is considered to be among the longest continuously recorded daily measurement made in science[81]. Many authors (see e.g.[64] and references therein) have already attempted to use neural networks to forecast aspects of the sunspot cycle, although as far as we are aware, most not in both space and time, having restricted themselves to using these neural networks to forecast mostly either the sunspot number or the sunspot areas time series. There is only one example[24], as far as we are aware, of actual spatial-temporal forecasts using neural networks (see also[82,83] where a neural network forecast of the magnetic flux, which is related to sunspots, is forecast for latitude/longitude datasets). There are also a few examples of forecasting the butterfly diagram sunspot data in both space and time (latitude/time)[22,84–88] but none of these used neural networks, rather all of those used other statistical methods or actual numerical physical modelling.

We take as a "training set" the data from the year 1874 to approximately 1997 (i.e. the first 1646 Carrington Rotations, from the number 275 to the number 1920). We attempt to reproduce or forecast the sunspot area butterfly pattern from Carrington Rotation 1921 up to 2076 (the last one corresponding approximately to the end of year 2008); in other words, we use 1646 time slices (≈ 122.92 years) to reproduce the next 156 time slices (≈ 11.57 years)[89]. The training set corresponds to around 12 solar cycles (cycle 11 to 22), while the "forecasting set" equates to cycle 23. The entire dataset, including the training and forecasting sets, is a grid $x^i_j = x(i,j)$, with $i = 1802$ and $j = 50$. The training set is a grid $x(1646,50)$. For this training set, the optimal values were $I^* = 2$, $J^* = 6$, $K^* = 9$ and $L^* = 70$, as calculated in[22]. **Notice that having some knowledge and understanding of the physics behind this data brings some light to these optimal values. In particular $L^* = 70$ corresponds to $L^*_{days} = 70 \times 27.2752316$ days $\approx 5.23 \approx 1/2$ solar cycle period.** The hyper-parameters of the neural network were: $N_h = 70$, $\eta = 0.3$, $\alpha = 0.01$, a logarithmic normalization of the inputs scaled with $\alpha_{nor} = 10$ and $\beta_{nor} = 0$, weight initialization with $\alpha_{rng} = 10^{-2}$ and $\beta_{rng} = -0.5$ and $N_{\text{steps}} = 1{,}000{,}000$. We used the logistic sigmoid function as the activation on both the hidden and output layers.

The Monte Carlo results are depicted in Fig. 3 showing runs with different I, J, K, L and plotting the SSIM versus the distance to the optimal input feature selection parameters (I^*,J^*,K^*,L^*) given by the dynamical systems theory. It shows a reasonable expected dispersion as proposed and a good convergence to the highest SSIM value we could obtain for this particular slicing of the training and forecast sets SSIM $= 0.836876152$. From the figure, there seems to be also two clusters of behaviour, and at closer inspection, we found that the cluster with lower SSIM is basically a set of very bad

forecasts, with none of the characteristics of the real sunspot behaviour (the 11 year-like cycle and the migration to the latitudinal equator), while the higher SSIM cluster corresponds to visually recognizable sunspot butterfly-like diagrams. **Furthermore, there are quite a few values of $d_e > 0$ that have a slightly higher SSIM than the SSIM of the supposedly optimal solution. These may be because first, this is not noiseless data, and therefore the results of the mutual information and the false nearest neighbours method may be out by a small amount, and second because it has been demonstrated that there can be a range of different delay values that give a good prediction. In fact it has recently been shown[90] that even with an embedding of slight lower dimension one can still obtain a good prediction. However, the overall qualitative aspects of our results in Fig. 3 are consistent with our proposal, within the constraints of the data we have.**

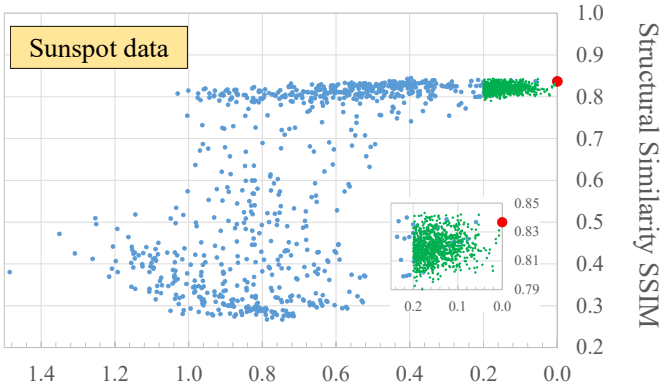


FIG. 3: Monte Carlo simulation of different input representations of the input layer for the neural network forecast for the sunspot data. It shows the structural similarity (SSIM) against how far (in a Euclidean space metric) the particular parameters of a particular run were from the supposedly optimal input representation parameters (red dot). **We have added (in green) all possible simulations with the distance smaller than 0.2, and did an overlay on top of the Monte Carlo randomly chosen scenarios.**

These results were quite satisfactory and inspired us to attempt to check the existence of a universality of behaviour across dynamical systems, by examining other unrelated synthetic generated data sets. We continue below to these attempts.

B. Coupled Hénon maps - a discrete-time dynamical system

Motivated by having a real case from a physical system, we then tried to investigate if this same proposal holds in a very simplified example of a spatial-temporal model. Coupled maps are widely used as models of spatial-temporal chaos and pattern/structure formation[91–93]. Following[17,18] we take a lattice of $M = 100$ coupled Hénon maps:

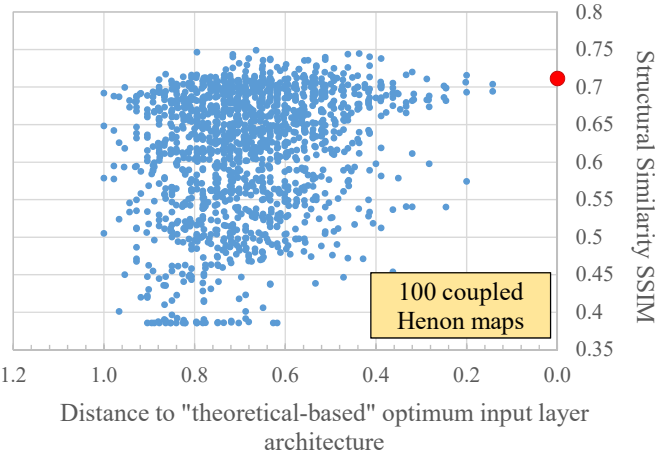


FIG. 4: Monte Carlo simulation of different input representations of the input layer for the neural network forecast for a series of 100 coupled Hénon maps. It shows the structural similarity (SSIM) against how far (in a Euclidean space metric) the particular parameters of a particular run was from the supposedly optimal input representation parameters (red dot). It also shows that as the parameters of a randomly chosen input representation get close to the supposedly optimal input representation ones, the SSIM converges to what seems to be the best possible forecast value given the limited dataset.

$$u_m^{n+1} = 1 - 1.45\left[\frac{1}{2}u_m^n + \frac{u_{m-1}^n + u_{m+1}^n}{4}\right]^2 + 0.3v_m^n, \quad (3)$$
$$v_m^{n+1} = u_m^n,$$

with fixed boundary conditions $u_1^n = u_M^n = \frac{1}{2}$ and $v_1^n = v_M^n = 0$. The initial values for rest of the variables $u_{m\neq 1,M}^{n=0}$ and $v_{m\neq 1,M}^{n=0}$ is taken from a random constant distribution in the range $[0,1[$.

We run the synthetic data generation for $N = 531$ time steps, and divided the set into $N_{\text{train}} = 500$ time steps for the training set and $N_{\text{test}} = 31$ time steps for the test set. The other parameters of the neural network were: $N_h = 10$, $\eta = 0.1$, $\alpha = 0$, a linear input normalization scaling with $\alpha_{nor} = 2.947992$, $\beta_{nor} = 0.515$, $\alpha_{rng} = 10^{-3}$, $\beta_{rng} = -0.5$ and $N_{\text{steps}} = 1{,}000{,}000$. We used the ReLu function as the activation on both the hidden and output layers.

For this case the optimal values given by the mutual information and the false neighbours methods were $I^* = 1$, $J^* = 3$, $K^* = 2$ and $L^* = 3$. The results of the Monte Carlo simulation for different I, J, K and L are depicted in Fig. 4. It again shows a dispersion as proposed and a reasonable convergence to the highest SSIM value we could obtain for this particular slicing of the training and forecast sets SSIM $= 0.71139101$. In this particular case, the optimal input layer architecture does not exactly lead to the highest value of the SSIM. We believe that this could be that, in the presence of limited data, we obtain a sub-optimal neural network training. Nonetheless and more importantly, the shape of the dispersion of the data in Fig. 4 is still overall consistent with our proposal.

Results suggest the same structure as depicted in our proposal diagram and in the previous results for sunspots. We now move below to a more complex model, a coupled set of ODEs.

C. Coupled Ordinary Differential Equations - Lorenz-96 model

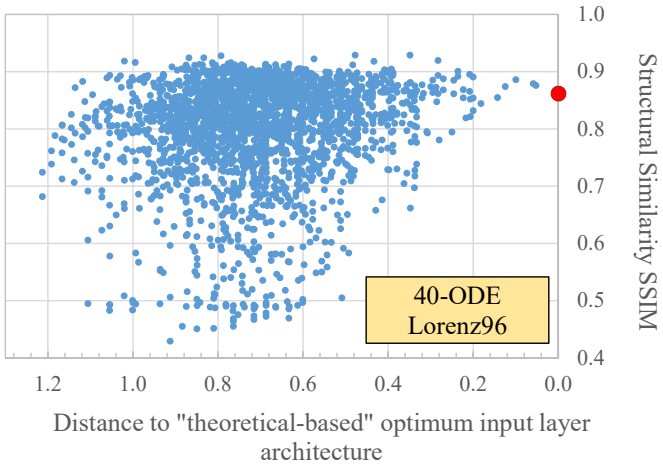


FIG. 5: Monte Carlo simulation of different input representations of the input layer for the neural network forecast for the 40-ODE Lorenz 96 system. It shows the structural similarity (SSIM) against how far (in a Euclidean space metric) the particular parameters of a particular run was from the supposedly optimal input representation parameters (red dot). It also shows that as the parameters of a randomly chosen input representation get close to the supposedly optimal input representation ones, the SSIM converges to what seems to be the best possible forecast value given the limited dataset.

For the spatially extended coupled ODEs model we used a well-known 40-coupled ODE dynamical system proposed by Edward Lorenz in 1996[94]:

$$\frac{dx_j}{dt} = \left(x_{j+1} - x_{j-2}\right) x_{j-1} - x_j + F,\ j = 1,\ldots,N = 40, \quad (4)$$

where $x_{-1} = x_{N-1}$, $x_0 = x_N$ and $x_{N+1} = x_1$ and F is a forcing term. We use the forcing $F = 5$ to get some interesting behaviour in space and time. We used a time step $\Delta t = 0.05$ and we have integrated this equation using J. Amezcua's MATLAB code as given in[95]. It uses the Runge-Kutta 4-step method.

We run the synthetic data generation for $N = 531$ time steps, and divided the set into $N_{\text{train}} = 500$ time steps for the training set and $N_{\text{test}} = 31$ time steps for the test set. The other parameters of the neural network were: $N_h = 10$, $\eta = 0.05$, $\alpha = 0.001$, a linear normalization input scaling with $\alpha_{nor} = 10$ and $\beta_{nor} = 0.430$, weight initialization with $\alpha_{rng} = 10^{-3}$ and $\beta_{rng} = -0.5$ and $N_{\text{steps}} = 100,000$. We used the ReLU function as the activation on both the hidden and output layers.

For this case the optimal values obtained before the Monte Carlo simulation from the mutual information and false neighbours methods were $I^* = 2$, $J^* = 2$, $K^* = 1$ and $L^* = 9$. The results of the random sampling of I, J, K, L in the simulation are depicted in Fig. 5. It shows a dispersion as proposed and a quite a good convergence to the highest SSIM value we could obtain for this particular slicing of the training and forecast sets: SSIM $= 0.861844038$. Results suggest the same structure as depicted in our proposal diagram and in the previous results for sunspots and the coupled Hénon maps. Again, in this particular case, the optimal input layer architecture does not exactly correspond to the highest value of the SSIM index. Again this could be that, in the presence of limited data, we only can get a sub-optimal network training. Nonetheless and more importantly, the shape of the dispersion of the data in Fig. 5 is still overall consistent with our proposal.

D. Partial Differential Equations - Kuramoto-Sivashinsky model

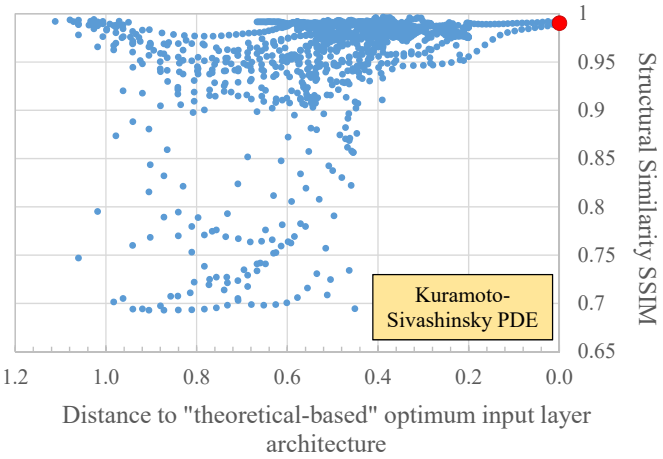


FIG. 6: Monte Carlo simulation of different input representations of the input layer for the neural network forecast for Kuramoto-Sivashinsky with $L = 22$ system. It shows the structural similarity (SSIM) against how far (in a Euclidean space metric) the particular parameters of a particular run was from the supposedly optimal input representation parameters (red dot). It also shows that as the parameters of a randomly chosen input representation get close to the supposedly optimal input representation ones, the SSIM converges to what seems to be the best possible forecast value given the limited dataset.

Finally we take a full PDE system, the Kuramoto-Sivashinsky model[96,97], a very well-known system capable of spatial-temporal chaos and complex spatial-temporal dynamics. It is a fourth-order nonlinear PDE introduced in the 1970s by Yoshiki Kuramoto and Gregory Sivashinsky to model the diffusive instabilities in a laminar flame front. The model is described by the following equation:

$$\frac{\partial u(x,t)}{\partial t} = -\frac{\partial^4 u(x,t)}{\partial x^4} - \frac{\partial^2 u(x,t)}{\partial x^2} - u(x,t)\frac{\partial u(x,t)}{\partial x}, \quad (5)$$

where $x \in [-\frac{L}{2}, +\frac{L}{2}]$ with a period boundary condition $u(x+L,t) = u(x,t)$. The nature of solutions depends on the system size L and on the initial $u(x,t=0)$. We have integrated this equation by taking an exponential time difference Runge-Kutta 4th order method (ETDRK4) using the Matlab code by P. Cvitanović as given in[98] and taking a time step of $\Delta t = 0.5$, $L = 22$ Fourier modes which are known to produce a "turbulent" or chaotic behaviour and a initial condition $u(x,t=0) = 10^{-5}$ for $x \in [5,15]$, the remain being $u(x,t=0) = 0$.

We run the simulation for $N = 531$ time steps, and divided the set into $N_{\text{train}} = 500$ time steps for the training set and $N_{\text{test}} = 31$ time steps for the test set. The other parameters of the neural network were: $N_h = 50$, $\eta = 0.1$, $\alpha = 0$, a linear normalization input scaling with $\alpha_{nor} = 5.8472$ and $\beta_{nor} = 0.5$, weight initialization with $\alpha_{rng} = 10^{-3}$ and $\beta_{rng} = -0.5$ and $N_{\text{steps}} = 1{,}000{,}000$. We used the ReLU function as the activation on both the hidden and output layers.

The results of the Monte Carlo simulation can be seen in Fig. 6. For this case the optimal values obtained before we run the Monte Carlo simulation were $I^* = 1$, $J^* = 2$, $K^* = 2$ and $L^* = 39$. It again shows a dispersion as proposed and a excellent convergence to the highest SSIM value we could obtain for this particular slicing of the training and test sets: a surprising high value of SSIM $= 0.990264382$. Results suggest the same structure as depicted in our proposal diagram and in the previous results for sunspots, the coupled Hénon maps and coupled ODEs.

V. COMPARATIVE APPROACHES

Here we analyse other approaches, first, other feature selection algorithms, and compare them to our non-linear embedding approach, and second, we examine other forecasting and machine learning models, and compare their performance against ours.

A. Other feature selection approaches

A valid question regarding our proposal is how do other well established feature selection approaches[99–102] compare with the one in our proposal. In the section we use a wide variety, by no means complete or exhaustive, of feature selection methods, and then run the neural network model with exactly the same hyper-parameters applied to exactly the same forecasting set as in section IV. In order to keep the number of calculations to within a reasonable size, we focus only on the most complex of the four data sets we introduced above, namely the sunspot data set introduced in section IV A. We start with a very large set of features, using the same architecture as in Fig. 1, but take the full feature set as represented by not only the values at the regular time-delay and space-delay crossings, but also all the intermediate values. That is we take the full set of $(2IK+1)(JL+1)$ features to forecast s_m^{n+1}. This is quite a high number of features to start with. For the sunspot data in section IV A, $(2IK+1)(JL+1) = 15{,}577$, a number of the same magnitude as the number of training vectors (17,150) that are possible to fit within the training set without going outside the boundary and much higher than the number of features selected by the non-linear embedding method from our proposal, which is $(2I+1)(J+1) = 35$.

Notice that in any forecast of temporal or spatial-temporal time series, the full number of features can be very high, in fact one can use the entire past history to forecast the future. This may lead to a accurate forecast on the training set, but one must remember that a high feature dimensionality can easily lead to over-fitting, and a bad forecast on the test set. This is another reason why it is important to test the effect of feature selection techniques and compare it against our non-linear embedding method proposal.

In order to compare like with like, for all the alternative feature selection methods, after applying then, i.e. after ranking or scoring, we take only the 35 top selected features to create the training set to pass to the neural network, so that we are in a equal footing with respect to the training set state vector dimensionality. We then calculate the SSIM index for each forecast to compare the relative strength of each feature selection approach against the one in our proposal. In addition to the SSIM index we also calculate a couple of other metrics which are relevant from the solar physics point of view in order to further differentiate the quality of each forecast comparatively to the target and to our own proposal's feature selection approach. The feature selection approaches we take are described below:

1. *Model based ranking*

In a model based ranking approach[99], a type of univariate feature selection, we select an arbitrary machine learning method, then build a forecasting model and measure the ranking or performance of each individual feature. Here we decided to use a random forest regressor[103], using the scikit-learn python framework[104] with a maximum depth equal to 4 levels and 20 estimators as our prediction method, and used the R^2 cross-validation score as our ranking metric. We used a shuffle split for the cross-validation element, with 5 splits and 25% as the proportion of the dataset to include in the test split. We then ranked the scores from higher to lower and select the top $(2I+1)(J+1) = 35$ features. Then this feature selection mask is implemented in a full run of the neural network as in section IV A.

2. *Linear Regression selection*

In this approach we used a plain vanilla linear regression using scikit-learn[100,104,105] and used the coefficients of the regression model to select and rank our features. This method assumes the highest ranking features will naturally have the

Feature Selection Approach	RMSE$\langle A(t)\rangle_{24}$	max $\left(\langle A(t)\rangle_{24}\right)$	SSIM $\left(A(t,\text{latitute}),A^*(t,\text{latitute})\right)$
Target original	0	1875	1
Non-linear embedding model	**451**	**1932**	**0.867534101**
Model based ranking	822	671	0.849603883
Linear Regression selection	1073	55	0.188479465
Lasso (L1) Regression selection $\alpha = 0.3$	1035	111	0.515152781
Lasso (L1) Regression selection $\alpha = 0.01$	1087	35	0.318327776
Lasso (L1) Regression selection $\alpha = 10$	1035	111	0.515152781
Ridge (L2) Regression selection $\alpha = 10$	1080	43	0.270791063
Mean decrease impurity	533	1329	0.838842336
Mean decrease impurity (Max Features = 35)	510	1329	0.857529382
Mean decrease accuracy	536	1373	0.847073739
Stability selection (Randomized Lasso)	1035	111	0.515152781
Recursive feature elimination	1085	37	0.155398607

TABLE I: Comparison of feature selection models, for forecasts using the neural network model and the physical data set $\left(A^*(t,\text{latitute})\right)$ as in section IV A. RMSE$\langle A(t)\rangle_{24}$ is the root mean square error of the difference of predicted $A(t)$ with respect to the target value of $A^*(t)$, with the forecasting period being the full solar cycle 23 (approximately 12 years long), averaged over 24 months and where $A(t) = \langle A(t,\theta)\rangle$ is the spatial (latitudinal) average of the sunspot areas $A(t,\theta)$. The max $\left(\langle A(t)\rangle_{24}\right)$ is the maximum of the 24-month smooth sunspot area cycle 23, used to compare if the forecast reaches the target sunspot area maxima. The SSIM and all neural network parameters are calculated and given as in section IV A.

highest (absolute) coefficients in the model, while the most irrelevant features will have (absolute) coefficients close to zero. As above, we selected the top $(2I+1)(J+1) = 35$ features. Again this feature selection mask is implemented in a full run of the neural network as in section IV A.

3. Lasso (L1) Regression selection

Here we do again a linear regression but with an L_1 regularization approach[100,104,106], that is commonly used to prevent overfitting and helps with model generalization. Technically a term of the following form $\alpha\Sigma_{i=1}^{N}\|w_i\|$ is added to the loss cost function, where α is a positive free parameter and w_i are the coefficients of the linear regression. The assumption here is that, as we have introduced a penalty factor that penalizes large model coefficients, the features which are irrelevant or of lower importance ranking will tend to have zero coefficients. It is well known that Lasso's regression tends to produce sparse solutions and therefore naturally leads to feature selection[107]. After ranking, we can then choose the most (35) relevant features and run the neural network with the feature selection mask as above.

4. Ridge (L2) Regression selection

Here we do again a linear regression but with an L_2 regularization approach[100,104,108], where we add a $\alpha\Sigma_{i=1}^{N}\|w_i\|^2$ to the loss cost function, where α is a positive free parameter and w_i are again the coefficients of the linear regression. The assumptions are similar to the ones above for the Lasso's regression approach, but the use of the L_2 norm tends to make the model more stable. After ranking, we can then choose the most relevant features (35 of them) and run the neural network with the feature selection mask as above.

5. Mean decrease impurity

In this approach we use a random forest as a regressor[101,103]. For regression trees, the measure (called impurity) used to decide locally the optimal condition is the variance. If we use a random forest, during training each aditional feature decreases the impurity and this can be averaged across the tree for our feature ranking. We then proceed as above.

6. Mean decrease accuracy

This approach measures the change that each feature has on the model accuracy, and again we use a random forest as a regressor[103]. This feature impact measurement is done via several permutations of the actual value of each feature and by measuring the accuracy changes. The assumption is that features which are irrelevant or close to irrelevant, the permutation will have little impact or consequence. We implement this using the approach as described in[101]. We then proceed as above.

7. Stability selection

This novel approach[102,109] uses subsampling with a selection method. After applying the approach on different subsamples of both the data and the features, it scores features

according to their impact/importance. We used the scikit-learn[104] implementation of stability selection in its randomized lasso implementation. We then proceed as above.

8. Recursive feature elimination

The recursive feature elimination approach[102,110] uses a regression model (in this case a linear regression) against a set of features, and recursively eliminates a portion of features based on e.g. the coefficients or scores, until there are no more features. Then it ranks the features according to the time these were eliminated. We used the recursive feature extraction (RFE) class within scikit-learn[104,111] to implement this feature selection method. We then proceed as above.

Our results are depicted in Table I. As mentioned above, we run the feature selection on all possible feature "boxes" of size $(2IK+1)(JL+1)$ =15,577 , then select the top $(2I+1)(J+1) = 35$ features, same as the non-linear embedding model in our proposal. The most important metric is the SSIM of the spatial-temporal prediction versus the target original cycle 23 (sunspot solar cycle). We can see that based on that metric alone our proposal's feature selection method is the best. However, some of the alternative approaches do get close to the value of SSIM from the non-linear model. Because of that, we introduced two other metrics. The first, $\max\left(\langle A(t)\rangle_{24}\right)$, measures the maximum amplitude of the time-smoothed cycle, and is one of the most compared metrics for forecasts of this data set in the literature. The second is the the root mean square error between the time-smoothed forecast versus the target (in two dimensions, space and time). We can clearly see that all other alternative feature selection approaches do not compare well against our non-linear embedding model when we look at all three metrics simultaneously.

B. Other forecasting models

Since the feature selection element of our proposal is actually independent of the neural network, we decided to also test it against other predictive models, to see if the proposal still holds for other forecasting methods. We tested four alternative models, a decision tree, a random forest, a variation of the AutoRegressive Integrated Moving Average (ARIMA) model, called Spatial-Temporal ARIMA or STARIMA, and finally a support vector machine (SVM) regression model. The details of these models are as follows.

1. Decision tree regression

Decision tree regression[112] is a simple approach, a variation of decision tree used for classification but where the predicted target is a real number value. The algorithm buckets the possible set of values within ranges, which act like target outcomes. We used the scikit-learn[104,113] implementation and left all settings as default straight out of the box except for the hyper-parameter representing the maximum depth of the tree, which we set as max_depth = 9.

2. Random forest regression

Random forests (also called random decision forests)[114] are used for regression by using an algorithm whereby a set of decision trees is fitted at training time and then the prediction is the mean prediction of that set of trees. This is a technique to avoid the overfitting that can be present if we use just one decision tree. It is a quite successful and well regarded method, used for both classification and regression. Here we used the scikit-learn[104,115] implementation. We set the random forest hyper-parameters as follows: the maximum depth of the tree max_depth = 9 and the number of trees in the forest n_estimators = 1.

3. STARIMA model

The STARIMA model[116] is a spatial-temporal extension of the Autoregressive integrated moving average (ARIMA) model[117]. In this case we take no integrated terms (we set them to zero), and we take the regressive terms to jump L steps in time, K steps in space, and take a fixed maximum of $J+1$ temporal terms and $2I+1$ spatial terms to forecast. We used the scikit-learn[104] implementation and the linear regression module[105] in scikit-learn.

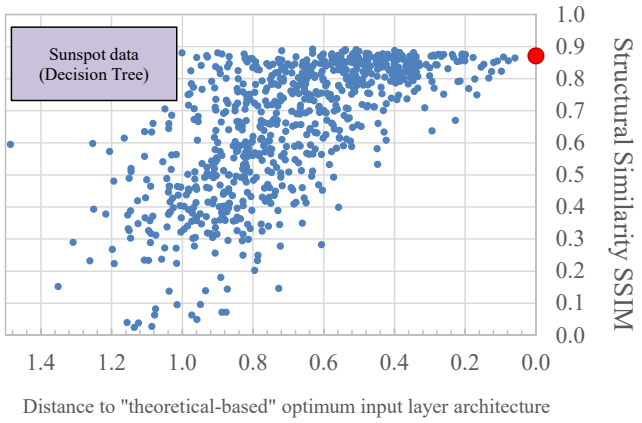
4. Support Vector Machine regression

Support Vector Machine regression[118] uses SVMs, whereby data is mapped in a way that different buckets (like categories) are separated by a gap as wide as possible. We used the scikit-learn[104,119] implementation and radial basis function (RBF) kernels for our approach. We set the random forest hyper-parameters as follows: the kernel coefficient gamma = 5, the penalty parameter of the error term $C = 1$ and the heuristic shrinking = FALSE. Because of memory/CPU limitations we run our Monte Carlo only with 100,000 examples of the training set, as opposed to 1,000,000 on all the other cases.

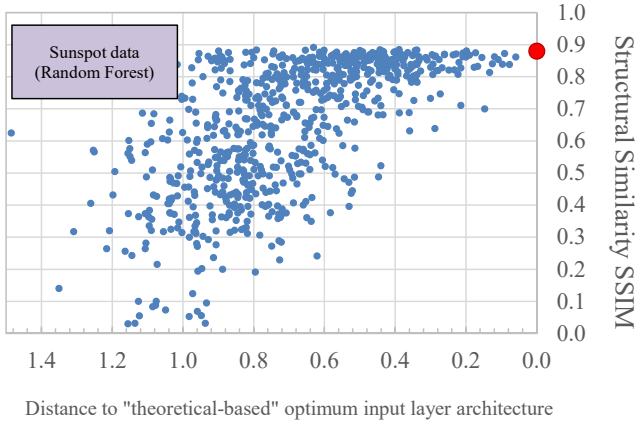
Again as in Section V A, we only take the sunspot data (the real data set) case, to keep the study within reasonable bounds. We take exactly the same Monte Carlo approach as in Section IV A, using the same set of I, J, K, L randomly chosen values. The results are depicted in Figure 7. This shows that the proposal holds for other forecasting models, implying that indeed this is a general result.

5. Deeper neural networks

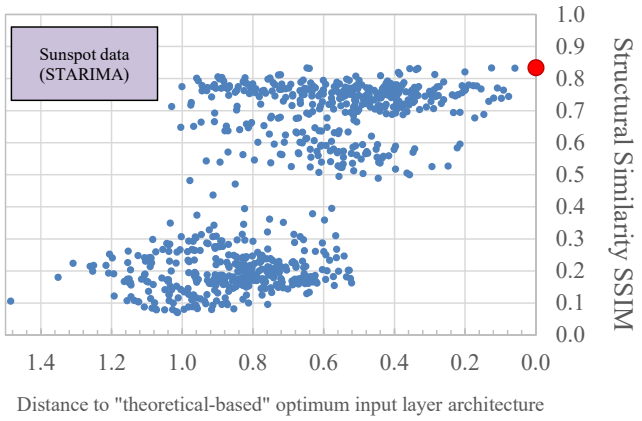
Here we assess if the proposal will hold when we relax the one hidden layer constraint we used in all neural network runs above. We run two Monte Carlo simulations, again on the



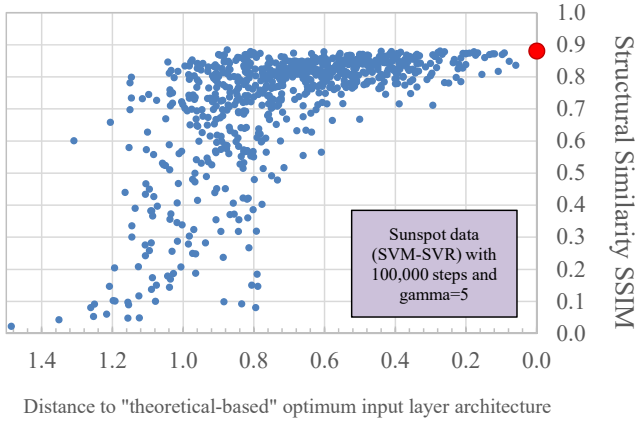
(a) Decision Tree Regression



(b) Random Forest Regression

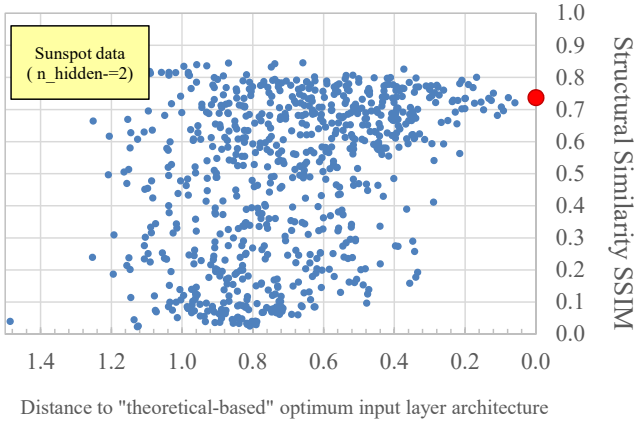


(c) Spatial-Temporal ARIMA

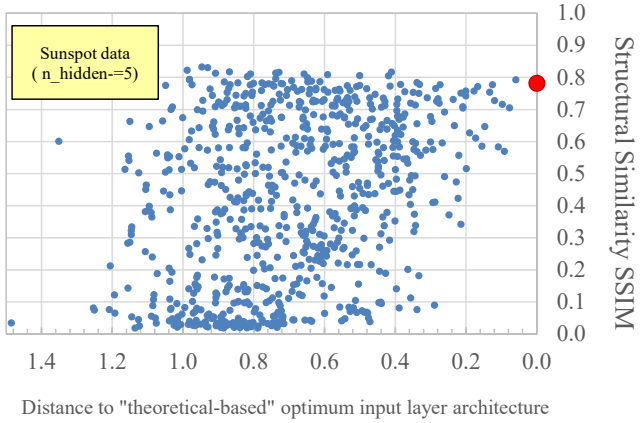


(d) SVM Support Vector regression

FIG. 7: Monte Carlo simulation of different feature selections for different regression model forecasts for the sunspot data. It shows the structural similarity (SSIM) against how far (in a Euclidean space metric) the particular parameters of a particular run were from the supposedly optimal input representation parameters (red dots).



(a) Number hidden layers = 2



(b) Number hidden layers = 5

FIG. 8: Monte Carlo simulation of different input representations of the input layer for the neural network forecast for the sunspot data for deeper networks, with 2 and 5 hidden layers. It shows the structural similarity (SSIM) against how far (in a Euclidean space metric) the particular parameters of a particular run were from the supposedly optimal input representation parameters (red dot).

sunspot data and this time with deeper feed-forward neural networks with 2 and 5 hidden layers. For the case with 2 hid-

den layers, we set the number of hidden nodes on the first layer as $n_{h_1} = 70$ and on the second layer $n_{h_2} = 60$, while for the case with 5 hidden layers we used $n_{h_1} = 70$, $n_{h_2} = 60$, $n_{h_3} = 50$, $n_{h_4} = 40$ and $n_{h_5} = 30$. These were chosen to ensure the number of nodes decreased while maintaining accuracy (as measured by the SSIM index).

As the back-propagation algorithm we developed is relatively slower on these deeper networks, we use the TensorFlow[120] python library with the Adam optimizer algorithm[121] for faster computation and convergence, after checking that the results and conclusions presented before did not change from moving from our own neural network code to TensorFlow code. The results are depicted in Figure 8 and again it shows that the proposal holds for other deeper models, implying that indeed this is a general result. We also noticed that the optimal input layer architecture does not exactly correspond to the highest value of the SSIM index. However, we believe this is because we maintained all the hyper-parameters (except the number of hidden layers, obviously) constant as in section IV A, and this leads to sub-optimal training. Nonetheless and more importantly, the shape of the dispersion in Fig. 8 is still overall consistent with our proposal.

VI. CONCLUSIONS

In this article, we have shown empirical evidence for the existence of an optimal feature selection for the input layer of feed-forward neural networks used to forecast spatial-temporal series. We believe that the selection of the features of the input layer can be uniquely determined by the data itself, using two techniques from dynamical systems embedding theory: the mutual information and the false neighbours methods. The former procedure determines the temporal and spatial delays to take when selecting features, while the latter determines the number of data points in space and time to be taken as inputs. We propose that this optimal feature selection gives the best forecast, as measured by a standard image similarity index. We also propose that the shape of the dispersion of points on a Monte Carlo simulation across all possible feature selections on a plot of the similarity index versus the distance to optimal feature selection is a skewed bell shape with the highest value being the optimal feature selection/maximum similarity index.

In order to substantiate our proposal, we chose four unrelated systems, in order of complexity: a set of spatially extended coupled maps; a set of spatially extended coupled ODEs; a one-dimensional spatial PDE and a real spatial-temporal data set from sunspots areas in our Sun. In all four cases, we were able to first use the mutual information and the false neighbours methods to determine the four parameters defining the input layer feature selection[122]. After calibration of the hyper-parameters we then were able to forecast the test set reasonably well, although, as explained, this was not the objective of this article. We show that for a random Monte Carlo simulation across possible feature selections, the neural network did not, as expected, forecast as well as it did for the specific set of optimal four parameters given by dynamical systems theory. As proposed, the Monte Carlo simulations show that the shape of the distribution of points was a skewed bell shape with the highest value being the optimal feature selection/maximum similarity index (subject to minor variations due to noise and the finiteness of the dataset). In order to further substantiate our proposal, we also compared our results against other alternative feature selection approaches and also the more complex case where we have a deeper neural network. In both cases, we have shown that the proposal holds and the non-linear dynamics approach for optimal feature selection is justified.

Given how important spatial-temporal systems are and how we want to forecast the future as accurately as possible it is quite important to attempt to reduce the number of hyper-parameters in neural network prediction, and to try to constrain the feature selection from the data properties only. If indeed our proposal turns out to be true, it would remove the input layer feature selection as another free parameter in the already complex process of choosing the details of the neural network to use for forecasting.

In this article we have focused first and foremost in establishing empirical evidence for our proposal, within a simple framework of feed-forward neural networks with one hidden layer for the purpose of prediction in one spatial and one temporal dimensions. Naturally, there are many clear extensions to our research. First to use deeper networks with additional hidden layers to possibly tackle systems which are hyper-chaotic (i.e. with multiple positive Lyapunov exponents). Second, to attempt to extend the proposal with empirical evidence in high dimensions, e.g. 3+1-dimensional weather systems. Third, to extend the proposal to other commonly used neural network models, such as recurrent neural networks[71], particularly echo state networks[25,123] and long short-term memory networks[124]. Fourth and last but not least, to demonstrate the proposal rigorously would show how dynamical systems theory can clarify the so called "dark art" in neural network feature construction. These objectives are however, outside the scope of this research article and will be pursued as part of future work.

ACKNOWLEDGMENTS

We would like to thank Prof. Reza Tavakol from Queen Mary University of London for very useful discussions on forecasting. We also thank Dr. David Hathaway from NASA's Ames Research Centre for providing the sunspot data on which some of the results in this article are based upon. CITEUC is funded by National Funds through FCT - Foundation for Science and Technology (project: UID/Multi/00611/2013) and FEDER - European Regional Development Fund through COMPETE 2020 – Operational Programme Competitiveness and Internationalization (project: POCI-01-0145-FEDER-006922). EB is supported by a UK RAEng Research Fellowship (RF/128).

[1] R. L. Devaney, *An Introduction To Chaotic Dynamical Systems* (CRC Press, 2018).

[2]P. Manneville, *INSTABILITIES, CHAOS AND TURBULENCE: An INTRODUCTION TO NONLINEAR DYNAMICS AND COMPLEX SYSTEMS. Edited by MANNEVILLE PAUL. Published by World Scientific Press, 2004. ISBN #9781860945335* (World Scientific Press, 2004).

[3]A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, Physica D Nonlinear Phenomena **16**, 285 (1985).

[4]H. Kantz, Physics Letters A **185**, 77 (1994).

[5]I. I. Gikhman and A. V. Skorokhod, *Introduction to the Theory of Random Processes (Dover Books on Mathematics)* (Dover Publications, 1996).

[6]J. Makhoul, Proceedings of the IEEE **63**, 561 (1975).

[7]M. J. D. Powell, in *Algorithms for Approximation*, edited by J. C. Mason and M. G. Cox (Clarendon Press, New York, NY, USA, 1987) pp. 143–167.

[8]D. S. Broomhead and D. Lowe, Complex Systems **2** (1988).

[9]J. D. Farmer and J. J. Sidorowich, Phys. Rev. Lett. **59**, 845 (1987).

[10]G. Box, G. M. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting & Control (3rd Edition)* (Prentice Hall, 1994).

[11]L. Rabiner and B. Juang, IEEE ASSP Magazine **3**, 4 (1986).

[12]M. Längkvist, L. Karlsson, and A. Loutfi, Pattern Recognition Letters **42**, 11 (2014).

[13]D. M. Rubin, Chaos: An Interdisciplinary Journal of Nonlinear Science **2**, 525 (1992).

[14]U. Parlitz and G. Mayer-Kress, Phys. Rev. E **51**, R2709 (1995).

[15]C. López, A. Álvarez, and E. Hernández-García, Phys. Rev. Lett. **85**, 2300 (2000).

[16]S. Ørstavik and J. Stark, Physics Letters A **247**, 145 (1998).

[17]U. Parlitz and C. Merkwirth, in *ESANN* (2000).

[18]U. Parlitz and C. Merkwirth, Physical Review Letters **84**, 1890 (2000).

[19]E. O. Covas and F. C. Mena, in *Dynamics, Games and Science I* (Springer Berlin Heidelberg, 2011) pp. 243–251.

[20]Y. Xia, H. Leung, and H. Chan, IEEE Transactions on Circuits and Systems II: Express Briefs **53**, 62 (2006).

[21]D. Gladish and C. Wikle, Environmetrics **25**, 230 (2014).

[22]E. Covas, A&A **605**, A44 (2017), arXiv:1709.02796 [astro-ph.SR].

[23]R. A. Richardson, Environmetrics **28**, e2456 (2017), e2456 env.2456.

[24]E. Covas, N. Peixinho, and J. Fernandes, Solar Physics **294**, 24 (2019).

[25]P. L. McDermott and C. K. Wikle, Stat **6**, 315 (2017).

[26]P. L. McDermott and C. K. Wikle, ArXiv e-prints (2017), arXiv:1711.00636 [stat.ME].

[27]M. Raissi, P. Perdikaris, and G. E. Karniadakis, ArXiv e-prints (2017), arXiv:1711.10566 [cs.AI].

[28]M. Raissi, P. Perdikaris, and G. E. Karniadakis, ArXiv e-prints (2017), arXiv:1711.10561 [cs.AI].

[29]Z. Long, Y. Lu, X. Ma, and B. Dong, ArXiv e-prints (2017), arXiv:1710.09668 [math.NA].

[30]J. Cao, D. J. Farnham, and U. Lall, ArXiv e-prints (2017), arXiv:1712.05293 [cs.LG].

[31]A. Ghaderi, B. M. Sanandaji, and F. Ghaderi, ArXiv e-prints (2017), arXiv:1707.08110.

[32]Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, Chaos **27**, 041102 (2017).

[33]M. Raissi and G. E. Karniadakis, Journal of Computational Physics **357**, 125 (2018), arXiv:1708.00588 [cs.AI].

[34]M. Raissi, ArXiv e-prints (2018), arXiv:1801.06637 [stat.ML].

[35]There is also a new emerging field of research on solving PDEs (therefore implicitly predicting a spatial-temporal evolution) using deep learning – see[? ? ?] and references therein. Furthermore, notice that in this article we are concerned with the full space-time prediction, as opposed to ongoing research on pattern recognition in moving images (2D and 3D), which attempt to pick particular features (e.g. car, pedestrian, bicycle, person, etc.) and to forecast where those features will be in subsequent images within a particular moving sequence – see[?] and reference therein.

[36]P. Lynch, *The Emergence of Numerical Weather Prediction: Richardson's Dream* (Cambridge University Press, 2006).

[37]E. N. Parker, *Cosmical Magnetic Fields: Their Origin and their Activity (The International Series of Monographs on Physics)* (Oxford University Press, 1979).

[38]M. West, D. Seaton, M. Dominique, D. Berghmans, B. Nicula, E. Pylyser, K. Stegen, and J. De Keyser, in *EGU General Assembly Conference Abstracts*, EGU General Assembly Conference Abstracts, Vol. 15 (2013) pp. EGU2013–10865.

[39]C. J. Schrijver, Space Weather **13**, 524 (2015), arXiv:1507.08730 [physics.space-ph].

[40]P. Grassberger, Physics Letters A **107**, 101 (1985).

[41]R. Bellman, *Dynamic Programming (Dover Books on Computer Science)* (Dover Publications, 2003).

[42]D. Cox and N. Pinto, in *Face and Gesture 2011* (2011) pp. 8–15.

[43]A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, in *Readings in Speech Recognition* (Elsevier, 1990) pp. 393–404.

[44]K. Luk, J. Ball, and A. Sharma, Journal of Hydrology **227**, 56 (2000).

[45]R. J. Frank, N. Davey, and S. P. Hunt, in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, Vol. 2 (2000) pp. 237–242 vol.2.

[46]R. J. Frank, N. Davey, and S. P. Hunt, Journal of Intelligent and Robotic Systems **31**, 91 (2001).

[47]K. Oh, Expert Systems with Applications **22**, 249 (2002).

[48]Z. Sheng, L. Hong-Xing, G. Dun-Tang, and D. Si-Dan, Chinese Physics **12**, 594 (2003).

[49]A. H. Ghaderi, B. Bharani, and H. Jalalkamali, International Journal of Modern Physics and Applications **1**, 64 (2015).

[50]F. Takens, Lecture Notes in Mathematics, Berlin Springer Verlag **898**, 366 (1981).

[51]H. Whitney, The Annals of Mathematics **37**, 645 (1936).

[52]R. Mañé, Lecture Notes in Mathematics, Berlin Springer Verlag **898**, 230 (1981).

[53]T. Sauer, J. A. Yorke, and M. Casdagli, Journal of Statistical Physics **65**, 579 (1991).

[54]Y. Gutman, ArXiv e-prints (2015), arXiv:1510.05843 [math.DS].

[55]Y. Gutman, Y. Qiao, and G. Szabó, Nonlinearity **31**, 597 (2018), arXiv:1708.05972 [math.DS].

[56]Notice that another non-linear dynamical systems technique exists to calculate this time delay, the zero of the autocorrelation function[58,59], but essentially these two approaches are after the same objective, i.e. to select uncorrelated variables as much as possible for optimal reconstruction embedding. So, in this article, we focus only on the first minima of the mutual information for simplicity of analysis.

[57]A. M. Fraser and H. L. Swinney, Physical Review A **33**, 1134 (1986).

[58]H. D. I. Abarbanel and J. P. Gollub, Phys. Today **49**, 86 (1996).

[59]H. Kantz and T. Schreiber, *Nonlinear time series analysis*, Cambridge nonlinear science series (Cambridge University Press, Cambridge, New York, 1997) originally published: 1997.

[60]M. B. Kennel, R. Brown, and H. D. I. Abarbanel, Phys. Rev. A **45**, 3403 (1992).

[61]J. M. Martinerie, A. M. Albano, A. I. Mees, and P. E. Rapp, Phys. Rev. A **45**, 7058 (1992).

[62]H. D. I. Abarbanel, R. Brown, J. J. Sidorowich, and L. S. Tsimring, Reviews of Modern Physics **65**, 1331 (1993).

[63]H. D. I. Abarbanel and J. P. Gollub, Phys. Today **49**, 86 (1996).

[64]R. Archana, A. Unnikrishnan, and R. Gopikakumari, in *2012 International Conference on Power, Signals, Controls and Computation* (IEEE, 2012).

[65]M. Ragulskis and K. Lukoseviciute, Neurocomputing **72**, 2618 (2009).

[66]F. Liu, C. Quek, and G. S. Ng, in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (IEEE, 2005).

[67]P. J. Werbos, in *System Modeling and Optimization* (Springer-Verlag, 2005) pp. 762–770.

[68]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Nature **323**, 533 (1986).

[69]P. Werbos, Proceedings of the IEEE **78**, 1550 (1990).

[70]Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).

[71]J. L. Elman, Cognitive Science **14**, 179 (1990).

[72]R. Reed, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks (A Bradford Book)* (A Bradford Book, 1999).

[73]Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, IEEE Transactions on Image Processing **13**, 600 (2004).

[74]It has also been used in the context of deep learning used for enhancing resolution on two dimensional images[?] and restoring missing data in images[?] .

[75]F. Bellot and E. E. Krause, The Mathematical Gazette **72**, 255 (1988).
[76]E. Covas, R. Tavakol, P. Ashwin, A. Tworkowski, and J. M. Brooke, Chaos **11**, 404 (2001), nlin/0105032.
[77]J. P. Boyd, *Chebyshev & Fourier Spectral Methods* (Springer Berlin Heidelberg, 1989).
[78]Given that the surface solar rotation varies with time and latitude, any approach of comparing positions on the Sun over a period of time is necessarily subjective. Therefore, solar rotation is arbitrarily taken to be 27.2752316 days for the purpose of Carrington rotations. Each solar rotation is given a number, the so-called Carrington Rotation Number, starting from 9th November, 1853.
[79]E. W. Maunder, MNRAS **64**, 747 (1904).
[80]"http://solarcyclescience.com/bin/bfly.jpg - (jpeg image, 8192 x 4358 pixels)," (2018), [Online; accessed 20. Apr. 2018].
[81]B. Owens, Nature **495**, 300 (2013).
[82]H. Lundstedt, M. Wik, and P. Wintoft, AGU Fall Meeting Abstracts , SH21A-0315 (2006).
[83]M. Wik, in *37th COSPAR Scientific Assembly*, COSPAR Meeting, Vol. 37 (2008) p. 3467.
[84]J. Jiang, R. H. Cameron, D. Schmitt, and M. Schüssler, A&A **528**, A82 (2011), arXiv:1102.1266 [astro-ph.SR].
[85]R. H. Cameron, J. Jiang, and M. Schüssler, ApJ **823**, L22 (2016), arXiv:1604.05405 [astro-ph.SR].
[86]S. W. McIntosh, X. Wang, R. J. Leamon, A. R. Davey, R. Howe, L. D. Krista, A. V. Malanushenko, R. S. Markel, J. W. Cirtain, J. B. Gurman, W. D. Pesnell, and M. J. Thompson, Astrophysical Journal **792** (2014), 10.1088/0004-637X/792/1/12, arXiv:1403.3071.
[87]J. Jiang and J. Cao, Journal of Atmospheric and Solar-Terrestrial Physics **176**, 34 (2018), arXiv:1707.00268 [astro-ph.SR].
[88]N. Safiullin, N. Kleeorin, S. Porshnev, I. Rogachevskii, and A. Ruzmaikin, Journal of Plasma Physics **84**, 735840306 (2018), arXiv:1712.07501 [astro-ph.SR].
[89]We use exactly the same data set as in[22,24] for consistency, even if more data is already available at this time.
[90]J. Garland and E. Bradley, Chaos: An Interdisciplinary Journal of Nonlinear Science **25**, 123108 (2015), https://doi.org/10.1063/1.4936242.
[91]K. Kaneko, Progress of Theoretical Physics Supplement **99**, 263 (1989).
[92]G. Mayer-Kress and K. Kaneko, Journal of Statistical Physics **54**, 1489 (1989).
[93]K. Kaneko, *Theory and Applications of Coupled Map Lattices (Nonlinear Science: Theory and Applications)* (Wiley, 1993).
[94]E. N. Lorenz, "Predictability - a problem partly solved," in *Predictability of Weather and Climate*, edited by T. Palmer and R. Hagedorn (Cambridge University Press, 2006) pp. 40–58.
[95]"https://www.mathworks.com/matlabcentral/fileexchange/25054-lorenz–96-model? - Lorenz '96 model - File Exchange - MATLAB Central," (2018), [Online; accessed 20. Apr. 2018].
[96]Y. Kuramoto and T. Tsuzuki, Progress of Theoretical Physics **55**, 356 (1976).
[97]G. I. Sivashinsky, Acta Astronautica **4**, 1177 (1977).
[98]"http://chaosbook.org/extras/KSEproject/html/index.html - Kuramoto-Sivashinsky: an investigation of spatiotemporal "turbulence"," (2007), [Online; accessed 20. Apr. 2018].
[99]"Feature selection – Part I: univariate selection | Diving into data," (2019), [Online; accessed 3. Jan. 2019].
[100]"Selecting good features – Part II: linear models and regularization | Diving into data," (2019), [Online; accessed 3. Jan. 2019].
[101]"Selecting good features – Part III: random forests | Diving into data," (2019), [Online; accessed 3. Jan. 2019].
[102]"Selecting good features – Part IV: stability selection, RFE and everything side by side | Diving into data," (2019), [Online; accessed 3. Jan. 2019].
[103]"3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.2 documentation," (2018), [Online; accessed 31. Dec. 2018].
[104]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).
[105]"sklearn.linear_model.LinearRegression — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 3. Jan. 2019].
[106]"sklearn.linear_model.Lasso — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 3. Jan. 2019].
[107]T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer New York, 2009).
[108]"sklearn.linear_model.Ridge — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 3. Jan. 2019].
[109]N. Meinshausen and P. Bühlmann, Journal of the Royal Statistical Society: Series B (Statistical Methodology) **72**, 417 (2010), https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2010.00740.x.
[110]I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, Machine Learning **46**, 389 (2002).
[111]"sklearn.feature_selection.RFE — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 3. Jan. 2019].
[112]A. D. Gordon, L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Biometrics **40**, 874 (1984).
[113]"Decision Tree Regression — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 4. Jan. 2019].
[114]A. Cutler, D. R. Cutler, and J. R. Stevens, in *Ensemble Machine Learning* (Springer US, 2012) pp. 157–175.
[115]"3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 4. Jan. 2019].
[116]P. E. Pfeifer and S. J. Deutsch, Transactions of the Institute of British Geographers **5**, 330 (1980).
[117]J. G. D. Gooijer and R. J. Hyndman, International Journal of Forecasting **22**, 443 (2006), twenty five years of forecasting.
[118]H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, in *Advances in Neural Information Processing Systems 9*, edited by M. C. Mozer, M. I. Jordan, and T. Petsche (MIT Press, 1997) pp. 155–161.
[119]"Support Vector Regression (SVR) using linear and non-linear kernels — scikit-learn 0.20.2 documentation," (2019), [Online; accessed 4. Jan. 2019].
[120]M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, and G. Brain, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)* (2016) arXiv:1605.08695.
[121]D. P. Kingma and J. Ba, arXiv e-prints , arXiv:1412.6980 (2014), arXiv:1412.6980 [cs.LG].
[122]We have four parameters for the feature selection in these cases, with one temporal and one spatial dimension. For higher dimensional systems, there will be more parameters, the exact number being double the number of dimensions of the system.
[123]E. Maiorino, F. M. Bianchi, L. Livi, A. Rizzi, and A. Sadeghian, Information Sciences **382-383**, 359 (2017).
[124]S. Hochreiter and J. Schmidhuber, Neural Computation **9**, 1735 (1997).