Prediction After a Horizon of Predictability: Non-Predictable Points and Partial Multi-Step Prediction for Chaotic Time Series

Vasilii A. Gromov, Philip S. Baranov, and Alexandr Yu. Tsybakin

School of Data Analysis and Artificial Intelligence

National Research University Higher School of Economics

Pokrovskii Boulevard, 11, Moscow, Russian Federation

Corresponding author: Vasilii A. Gromov, stroller@rambler.ru

Abstract. The paper discusses several novel strategies for multi-step prediction of chaotic time series. Generalized z-vectors (irregular embeddings), comprising non-successive observations, make it possible to obtain for each point to be predicted, a fairly large set of possible predicted values. If one examines such a set, one may ascertain whether it is possible to produce a unified predicted value for it or not (whether the point is predictable or non-predictable), and determine this unified value, if it exists. With non-predictable points, one may state the partial multi-step prediction problem as a two-objective problem. The first functional minimizes the number of non-predictable points, the second, average error for predictable ones. The main difference of strategies designed for such statements from their classical counterparts is non-predictable points and, consequently, an ability not to account for predictions at intermediate positions that are clearly erroneous. It appears that for such algorithms the number of non-predictable points grows exponentially with a prediction horizon, but an average error for predictable points remains constant and rather small up to a horizon of predictability and even farther for benchmark and real-world time series.

Keywords. Chaotic time series, multi-step prediction, predictive clustering, predictable and non-predictable points, a horizon of predictability.

1. Introduction

Chaotic systems, both social and natural, have a widespread currency. This may explain a constantly increasing number of prediction algorithms for chaotic time series. It is worth stressing however, that while the algorithms for a single-step-ahead prediction [2] appear to be amazingly efficient; their counterparts for a multi-step ahead (*MSA*) prediction are still in their infancy. One may attribute

this fact to an exponential growth of an average prediction error with a prediction horizon, that is the number of steps ahead to be predicted.

This exponential growth reflects Lyapunov instability, immanent to any chaotic system. Namely, according to Lyapunov instability definition, however small is any initial difference between two neighbouring trajectories, the difference grows exponentially with time, and its exponent equals to the highest Lyapunov exponent [3, 4]. Laypunov instability also leads to another concept, that of a horizon of predictability (sometimes also referred to as Lyapunov time [5]). For a given observational error $\varepsilon(0)$ and the maximum prediction error ε_{max}, the exponential growth mentioned above satisfies $\varepsilon(T)(0)^{\lambda T}{}_{max}$, thereby giving a horizon of predictability $T \sim \frac{1}{\lambda} ln \frac{\varepsilon_{max}}{\varepsilon(0)}$ [3, 4]. In the context of a multi-step ahead prediction, this implies that the smallest difference between true and predicted values in an intermediate point (that is the point between the last observable point and the point to be predicted) – inevitable for any prediction method – triggers exponential error growth in all subsequent intermediate points and in the point to be predicted itself. Therefore, a horizon of predictability is an upper theoretical boundary for the number of steps ahead to be predicted for any prediction algorithm (for given $\varepsilon(0)$ and ε_{max}). A horizon of predictability should not be confused with a prediction horizon, which is a mere number of steps ahead the algorithm makes its prediction. Usually, a prediction horizon h is essentially lesser than a horizon of predictability T: $T >> h$.

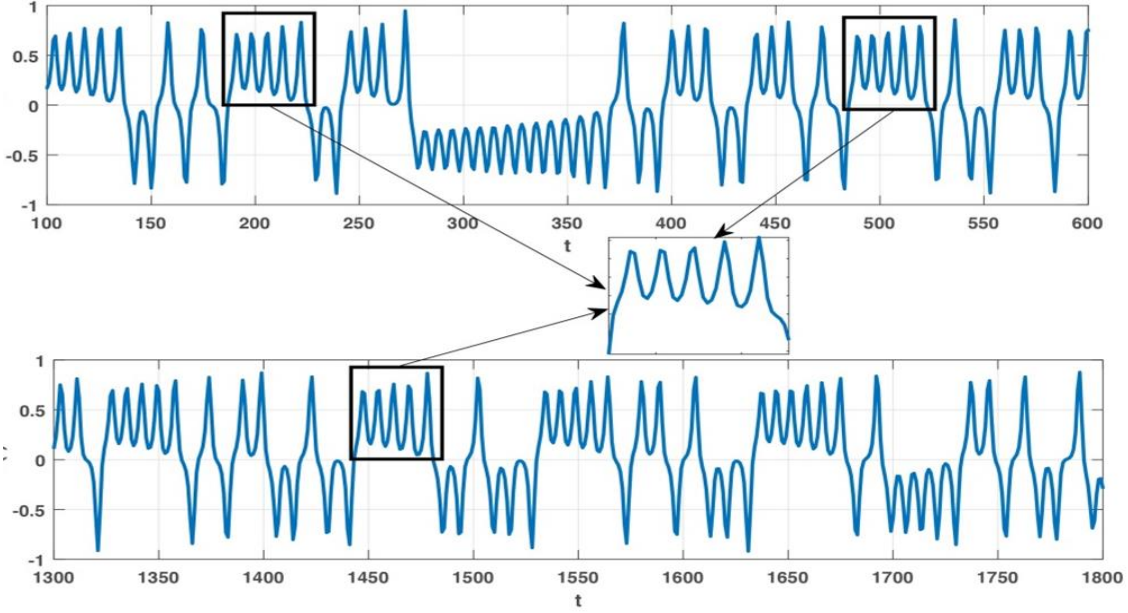


Fig. 1. Diagrammatical representation the way the algorithm searches for similar series sections (the top subfigure), obtains a cluster center (an inset in the middle), and predicts when finds similar section in a testing set (the down subfigure)

Nonetheless, this apparently inevitable exponential growth only seems to be so. Predictive clustering [6, 7] furnishes several tools that, applied together, make it possible to avoid it.

First, predictive clustering as such uses motifs, typical sequences that occur repeatedly in a series with small variations. In a prediction procedure, if a section of a series appears to be close enough to an initial section of some motif, then it is highly likely that subsequent observations will be close to the motif final section. Therefore, points of this final section are used as predicted values for the subsequent observations. In order to obtain motifs, one clusters vectors that comprise successive observations of a series in question (z-vectors) (irregular embeddings [1]); motifs are centres of the clusters. It is worth stressing that the majority of prediction methods attempt to construct a unified, global prediction model, whereas predictive clustering methods, vice versa, build up a set of separate, local prediction models, motifs (cf. global vs. local learning [10, 11]).

Second, prediction clustering allows one to develop the concept of non-predictable points in a quite natural way [7]. A non-predictable point means a point such that the algorithm failed to find a motif close enough and hence has to predict it in a proper way. It is worthy to emphasize that if one introduces the concept of non-predictable points (non-predictables), one thereby renders the problem two-objective: one must minimize

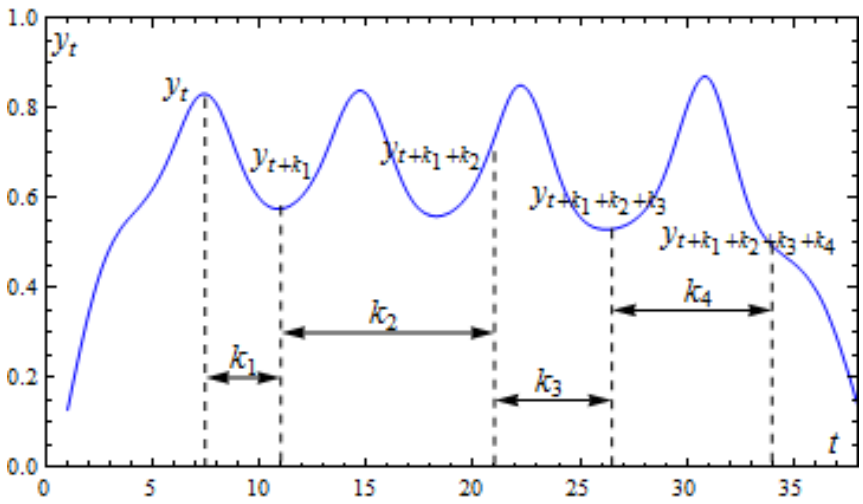


Fig. 2. A sample vector $(y_t, y_{t+k_1}, \ldots, y_{t+k_1+\ldots+k_{L-1}})$ concatenated for a pattern $k_1, \ldots, k_{L-1}, k_i \in N$

simultaneously the number of non-predictable points and an average error for predictable ones. A classical statement of a prediction problem is that the first criterion is disregarded. We feel this is a great advantage, if an algorithm is capable of identifying non-predictable points: it is much better if the algorithm 'honestly' notifies that it is unable to predict a point than it predicts it anyway regardless of how accurate or not the prediction is. For some tasks (for example, for financial trading) such a statement sounds natural: nobody compels a trader to sell or buy at each instant of time; on the contrary, he may work only with points that the algorithm recognizes as predictable. In the context of dynamical systems [3, 4], each cluster (by definition, it includes similar sections of a trajectory) corresponds to a particular area of a strange attractor. Furthermore, areas with high values of an invariant measure are associated with large clusters, clusters corresponding to frequently observed motifs, whereas areas with little values, to small ones, ones corresponding to rarely observed motifs. This means that as a training set increases, so does the number of clusters. The number of non-predictable points and the average error for predictable ones, on the contrary, decreases. A large-scale simulation for the Lorenz series [11] corroborates this conclusion.

Third, Gromov and Borisenko [7] introduced generalized z-vectors, that is, z-vectors comprised of non-successive observations according to certain predefined patterns. A pattern is defined as a pre-set sequence of distances between positions of observations, such that these (non-successive) observations are to be placed on the successive positions in a newly generated sample vector. The vector, thus concatenated, generalizes a conventional z-vector [3, 4], which corresponds to the pattern $(1,1,..,1)$ $(1,1,...,1)(n\,L-1$ times). Figure 1 presents schematically, a sample vector constructed according to the pattern of $k_1,...,k_{L-1}, k_i \in N$ (Fig. 1), the way the vectors are clustered to produce a motif, and the way the respective motif is used to predict the time series at hand (Fig. 2). A reader may visualize that, in order to construct a vector according to this pattern, we place against a series at hand, a comb with all teeth, but with L broken away in such a way that distances between 'extant' teeth equal to $k_1,...,k_{L-1}$, respectively. Moving the comb along the series, we, for each position t, obtain a vector of observations, to be under the teeth of the comb, $\left(y_t, y_{t+k_1}, ..., y_{t+k_1+...+k_{L-1}}\right)$. To clarify, for a given pattern, a conventional sliding window is replaced by a sliding comb with the majority of its teeth broken off. A set of such vectors comprises a sample corresponding to a pattern of $k_1,...,k_{L-1}, k_i \in N$. Each such sample is clustered separately. In terms of strategies of multi-step ahead prediction, this strategy combines two conventional strategies, namely, the iterative and direct: it employs predicted values to predict farther as the iterative does, and it uses different predictions for the same point (a set of possible

prediction values) as the direct does. It is worth noting that since the distance between the first and last position for some patterns is quite large, the exponential growth, peculiar to iterative strategy, is somewhat alleviated.

Fourth, since the number of patterns is rather large, so is the number of possible predicted values that can be obtained for each position; though these values are not always statistically independent. This makes it possible to design a great deal of various algorithms to determine a unified prediction value for a set of possible prediction values (for instance, by averaging them). On the other hand – and most importantly – it is possible to extend the concept of the non-predictable points. Namely, besides points that are non-predictable on the basis that there are no motifs close enough to make predictions, a set of non-predictable points now includes points for which it is impossible to determine a unified prediction value for a set of its possible prediction values. A frequent case of a possible prediction values set that consists of two equinumerous clusters with distinctly different centres, exemplifies such non-predictable points.

It is worth noting that owing to patterns with a distance between penultimate and ultimate teeth greater than 1, $k_{L-1} > 1$, the fact that a position t is non-predictable does not imply that positions $t + 1$, $t + 2$, etc. would be non-predictable too. This fact is of fundamental importance for multi-step ahead prediction: If one needs to predict a point at a position $t + h$, one employs the iterative strategy (or rather its modification based on patterns of non-successive observations), namely, one predicts values for intermediate positions between t and $t + h$ step by step, identify non-predictable points in the course of it, and step over them. This results in non-predictable sections in between predictable positions. We used to refer to the former as ‘quagmire’, to the latter as ‘tussocks’. Then the entire prediction process is likened to jumping from a tussock to a tussock in order to negotiate a swamp, or to put it differently, using stepping stones.

It is impossible here to obtain a prediction for each point (this explains the word ‘partial’ in the title), but such a strategy allows one to predict to a prediction horizon (to the number of steps ahead) comparable with a horizon of predictability, or even more than it. The present paper discusses several methods of identifying non-predictable points and corresponding strategies of multi-step ahead prediction; they made it possible to partially predict benchmark and real-world time series to a horizon of predictability and farther.

The rest of the paper is organized as follows. The next section reviews recent advances in the field; the third section formally states the problems under study; the fourth outlines clustering method, and methods employed to obtain predictions and also identify non-predictable points; the fifth section provides the prediction results for a time series generated by the Lorenz system, and a time

series of hourly load values in Germany. Finally, the last section presents conclusions.

2. Related works

Recent papers that concern themselves with chaotic time series prediction are quite numerous [2, 7, 10, 12-14], but the vast majority of them attack a one-step-ahead prediction problem, along various lines. Meanwhile, papers dealing with the multi-step ahead (*MSA*) prediction are less numerous. Perhaps, this fact is attributable to the (discussed above) exponential growth of an average prediction error as a function of a prediction horizon.

An MSA prediction algorithm for chaotic time series usually contains two constituents, which are responsible, in the end, for prediction quality. The first is a technique used for a one- or few-step-ahead prediction. The second is a strategy utilized to 'assemble' one- or few-step ahead predictions to a multi-step ahead prediction.

Algorithms used for a one-step-ahead prediction cover nearly all fields of data mining and machine learning: support vector regression and its modifications [12], multilayer perceptron [15, 16], cluster centres in predictive clustering [8, 17-19], LSTM neural networks [13], Voronoi regions partitioning [20], ridge polynomial neural networks [21], wavelet neural networks [22], dilated convolution networks [23], to name just a few. One may construe dilated convolution networks as counterparts of patterns of non-successive observations (see below).

Another factor of fundamental importance is the strategies of MSA prediction. Basic ones are the iterated (recursive) and direct [9, 24] strategies. The first strategy implies that multi-step ahead prediction is carried out step by step, using predicted values from previous steps. The second strategy suggests obtaining such results immediately, without making predictions for intermediate positions. The strategy applies prediction techniques for various prediction horizons, thereby providing multiple predictions for a position to be predicted. Taieb, Sorjamaa, and Bontempi [10] review methods based on the two basic strategies. Sangiorgio and Dercole [13] apply both strategies with multilayer perceptrons and LSTM nets employed as tools to predict for a one step ahead.

Unfortunately, MSA methods designed in the framework of the above strategies are, in one way or another, amenable to 'a curse of an exponential growth' discussed above. This gives rise to a number of hybrid strategies aiming to solve this problem. Ben Taieb et al. [9], in their brilliant review, compare two basic and three novel strategies (DirRec, MIMO, and DIRMO). DirRec (=Direct+Recursive) combines two basic strategies to the effect that it uses the direct approach to predict values, but the number of inputs is enlarged iteratively to include values for just predicted positions. MIMO (multiple-input multiple-

output strategy) [25] implies that the algorithm predicts not a single value corresponding to a prediction horizon, but an array of values simultaneously, for positions between the last observed values and the horizon inclusive. This makes it possible to retain in predicted values, correlations present in any time series considered. DIRMO (=DIRrec+miMO) implies to divide positions, up to and including a prediction horizon, into blocks, and to apply MIMO strategy to each block separately. Authors test the five strategies on a reasonably broad sample of time series (competition NN5), which reflects various irregularities inherent to time series.

Bao, Xiong, and Hu [12] examine comparative efficiency of the three strategies (iterated, direct, and MIMO); a one-step-ahead prediction is performed with a modified support vector regression. According to the authors, all other factors being equal, MIMO compares favourably with two other strategies. Multiple-task learning can be viewed as a kind of strategy for multi-step ahead prediction. Chandra, Ong, and Goh [15] propose an algorithm to determine a neural network structure to solve MSA prediction problems; the approach can be considered as a combination of the direct and iterated strategies. Wang et al. [26] utilize a neural model in order to combine periodic approximations for long periods and machine learning models for short periods. One can view this approach as a kind of strategy to cope with data with a marked periodicity. Ye and Dai [14] employ multi-task learning for multi-step prediction; it is possible to view multi-task learning as a kind of MSA predicted strategy. Kurogi, Shigematsu, and Ono ([20]) make use of an out-of-bag model to select models for multi-step prediction. Authors aim at predicting a chaotic series as far as it is possible (with the maximum prediction horizon) while retaining reasonable accuracy. Authors present results for the Lorenz series. The present paper discusses a few novel strategies; the main difference from their classical counterparts is non-predictable points, and consequently, an ability to not account for predictions at intermediate positions that are clearly erroneous [7].

3. Problem statement

The present paper considers a h-step-ahead prediction problem for a chaotic time series $Y = \{y_0, y_1, \ldots, y_t, \ldots\}$, where h is an integer, strictly greater than zero, $h \in N, h > 0$. We assume that all transient processes in the system that generate the time series at hand (the system is usually unknown) have been completed, and the time series reflects the trajectory movement in the neighbourhood of a strange attractor, however complex it can be. The second assumption is that the series meets Takens theorem conditions and, respectively, one can analyse the attractor structure using time series observations [3, 4].

A series is divided into $Y = Y_1 \cup Y_2$, where $Y_1 = Y_1(t) = \{y_0, y_1, \ldots, y_t\}$ is an observable part, used to train a prediction model(s), (a training set) and Y_2 is an unknown part, used to test them (a test set). $Y_1 \cap Y_2 = \emptyset$. When a prediction algorithm is tested to predict a value at a position $t + h$ of a test set, it possesses information about observations from $t - s + 1$ to t inclusive, but *does not* possess information observations from $t + 1$ to $t + h - 1$ inclusive, $\hat{y}_{t+h} = \hat{y}_{t+h}(\{y_t, y_{t-1}, \ldots, y_{t-s+1}\})$. s is a parameter of the algorithm.

Predictive-clustering algorithms and a large number of used patterns make it possible to build up a set of possible predicted values $\hat{S}_{t+h} = \left\{\hat{y}_{t+h}^{(1)}, \ldots, \hat{y}_{t+h}^{(N_{t+h})}\right\}$ for each position to be predicted. N_{t+h} is the number of possible predicted values found by an algorithm, $\hat{y}_{t+h}^{(i)}, i = 1..N_{t+h}$ is the ith predicted value. A set $\hat{S}_{t+h}^{(p)}$ is defined as $\hat{S}_{t+h}^{(p)} = \{\hat{S}_{t+h}, \ldots, \hat{S}_{t+h-p+1}\}$ and comprises sets of possible predicted values for the position $t + h$ itself and for $p - 1$ preceding positions. p is a parameter of the algorithm. It usually takes values $p = 1$ or $p = h$. For the first case, the set $\hat{S}_{t+h}^{(p)}$ consists of the set of possible predicted values for the position $t + h$ only, for the second, for the position $t + h$ itself and all intermediate positions between the last observable point and this position. We denote an algorithm that builds up sets of possible predicted values $\hat{S}_{t+h}$ as f_h: $\hat{S}_{t+h} = f_h(\{y_t, \ldots, y_{t-s+1}\})$.

The concept of non-predictable points implies that one applies two operators to the set $\hat{S}_{t+h}^{(p)}$. The first operator checks whether the position is predictable:

$$\zeta\left(\hat{S}_{t+h}^{(p)}\right) = \begin{cases} 1, if\ \ position\ \ ispredictable \\ 0, otherwise \end{cases} \quad (1)$$

where $\zeta(\emptyset) = 0$ for any function ζ.

The second determines a unified predicted value for a set of possible predicted values, provided the position appears to be predictable:

$$\hat{y}_{t+h} = g\left(\hat{S}_{t+h}^{(p)}\right) \quad (2)$$

In these terms, a multi-step ahead prediction problem is stated as a two-objective optimization problem:

$$I_1 = min \sum_{t+h \in Y_2} \left(1 - \zeta\left(\hat{S}_{t+h}^{(p)}\right)\right) \quad (3)$$

$$I_2 = min \frac{1}{|Y_2|} \sum_{t+h \in Y_2} \zeta\left(\hat{S}_{t+h}^{(p)}\right) \left\| g\left(\hat{S}_{t+h}^{(p)}\right) - y_{t+h} \right\|^2 \quad (4)$$

For both cases, one sums over all observations of a test set. The first functional minimizes the number of non-predictable points, the second, average error for predictable ones. In the present paper, authors attempt to solve the two-objective optimization problem (3), (4).

4. Prediction algorithm

In this section, we describe the proposed prediction algorithm. The first subsection deals with a way to generate samples from the time series at hand according to predefined patterns; the second, with a clustering technique employed. The third, fourth, and fifth subsections discuss possible ways to generate a set of possible prediction values, quality measures for algorithms to identify non-predictable points, and techniques used to identify such points, respectively. In the present section, the description of any technique is finished with a symbolic label in square brackets, used to designate it in the subsequent section, **Numerical Results**.

2.1. A training set

The series is considered to be normalized. To generate samples, we use a concept of a pattern. Mathematically, a pattern is defined as a pre-set array of distances between positions of time series observations, such that these (non-successive) observations are to be placed on the successive positions in a sample vector. Thus, each pattern is a $L-1$-dimensional integer vector $(k_1, k_2, \ldots, k_{L-1})$, $1 \le k_j \le K_{max}$; the parameter K_{max} dictates the maximum distance between positions of observations that become successive in the vector to be generated. Thereby, the quantity $(L-1)K_{max}$ symbolizes a kind of memory depth. $\aleph(L, K_{max}())$ denotes a set of all possible patterns of the specified length L. The vector, thus concatenated, generalizes a conventional z-vector [3, 4], which corresponds to the pattern (1, 1, …, 1) ($L-1$ times).

For example, let us consider a four-point pattern (2, 3, 4). For this pattern, the two first vectors of a training set are (y_0, y_2, y_5, y_9) and (y_1, y_3, y_6, y_{10}); the last one is $(y_{t-9}, y_{t-7}, y_{t-4}, y_t)$, where y_t is the last observable value.

Conventionally, one uses, for predictive clustering, vectors concatenated successive observations (z-vectors); they prove less efficient than those based on the vectors concatenated according to various patterns, generalized z-vectors [7]. One may attribute this to the fact that vectors of non-successive observations are able to store information about salient observations (minima, maxima, tipping points, and so forth) and correlations between them. Figure 2 above, symbolically shows a pattern superimposed on a time series in order to generate a sample vector.

In order to generate training sets, one may employ the entire set of all combinatorically possible patterns $\aleph(L, K_{max}())$ or the part thereof consisting of randomly chosen patterns. The patterns should be chosen before simulation starts. A set of used patterns is denoted as $\beta\aleph(L, K_{max}())$, where β is a percentage of patterns used. In particular, $100\%\aleph(L, K_{max}())$ corresponds to the case when all possible patterns are used.

Training sets are generated independently for each pattern $\alpha \in \beta\aleph(L, K_{max}())$. Motifs may be extracted from each training set by two possible ways. The first one utilizes vectors of a training set as motifs [*p*] (pointwise); this approach bears a close resemblance to lazy learning [9]. The second implies that a sample is clustered, and centres of the clusters are used as the motifs in question [*cl*] (cluster); except as otherwise noted, the clustering technique is the modified Wishart (please, refer to the next subsection). The first method, despite all drawbacks, compares favourably with the first in terms of required computations.

We denote a set of motifs corresponding to a pattern $\alpha = \left(k_1^{(\alpha)}, \ldots, k_{L-1}^{(\alpha)}\right)$, $\alpha \in \beta\aleph(L, K_{max}())$ as $\Psi_\alpha = \{C_\alpha\}$, $C_\alpha = \left(\eta_1^{(\alpha)}, \ldots, \eta_{L-1}^{(\alpha)}\right)$. One should emphasize that motifs thus obtained, can be used only with the respective patterns α, which indicates distances between positions in a series, such that values $\eta_1^{(\alpha)}, \ldots, \eta_{L-1}^{(\alpha)}$ may occupy. A set of all used motifs is denoted as $\Psi = \{(\alpha, \Psi_\alpha)\}$, $\alpha \in \beta\aleph(L, K_{max}())$.

2.2. Clustering technique

As the trajectory of a system moves along the same area of the attractor frequently, one can meet similar sequences in the time series associated with that area. If one reveals these areas, describes them, and develops the simplest prediction models for each area, one makes it possible to predict chaotic time series up to a considerable time limit [17]. The clustering method presented below is employed to collect sequences belonging to the same cluster.

To cluster vectors, we employ the Wishart clustering technique [27, 28] as modified by Lapko and Chentsov [29]. The method employs graph theory

concepts and a non-parametric probability density function estimator of r -nearest neighbours. Some difficulties associated with the application of the algorithm to forecasting problems are discussed in [7].

An estimated saliency for a point x is defined as $p(x) = \frac{r}{V_r(x)n}$, where $V_r(x)$ and $d_r(x)$ are, respectively, a volume and a radius of the minimum hypersphere with its centre at point x containing at least r observations. The method relies upon a similarity relation graph $G(Z_n, U_n)$; its vertices correspond to samples and edges defined as $U_n = \{(x_i, x_j): d(x_i, x_j) \le d_r(x_i), i \ne j\}$. $G(Z_q, U_q)$ is a generated subgraph for graph $G(Z_n, U_n)$, with a vertex set $Z_q = \{x_j, j = 1..q\}$ and an edge set that comprises all edges from U_n, such that their final vertices belong to the set Z_q. $w(x_q)$ is a cluster number (label) for x_q. The cluster c_l ($l > 0$) is defined to be a height-significant one, with respect to height value $\mu > 0$ if $\max_{x_i, x_j \in c_l} \{|p(x_i) - p(x_j)|\} \ge \mu$.

Thus, the algorithm consists of the following steps:

1. Determine distances $d_r(x_q)$ for each sample to its r -nearest neighbour and to range the samples in ascending order of the distances calculated.
2. Set $q = 1$.
3. For subgraph $G(Z_q, U_q)$ the following alternatives are possible:
   - If a vertex x_q is an isolated vertex of $G(Z_q, U_q)$, then generate a new cluster.
   - If a vertex x_q is connected to vertices of a lth cluster only and the cluster is completed, then set $w(x_q) = 0$. Otherwise (if the cluster is not completed yet) set $w(x_q) = l$.
   - If a vertex x_q is connected to vertices of clusters $l_1, l_2, \ldots, l_a, a > 1$.
   - If all a clusters are completed, then to set $w(x_q) = 0$.
   - Otherwise, determine $\kappa(\mu) \le a$, a number of significant clusters.
     - If $\kappa(\mu) > 1$ or $l_1 = 0$, then set $w(x_q) = 0$, label significant clusters as completed, and delete insignificant clusters, setting $w(x_q) = 0$ for all samples that belong to them.
     - Otherwise, incorporate clusters $l_2, \ldots, l_a$ to l_1, setting $w(x_q) = l_1$ for all samples belonging to them and for the sample itself.
4. Set $q = q + 1$. If $q \le n$, then go to step 3.

To be specific, we used the following values: $r = 11$, $\mu = 0.2$.

2.3. Sets of possible predicted values ($\hat{S}_{t+h}^{(p)}$)

For a given pattern $\alpha \in \beta\aleph(L,K_{max})$, $\alpha = \left(k_1^{(\alpha)},\ldots,k_{L-1}^{(\alpha)}\right)$, one considers a set of all corresponding motifs Ξ_α and, using them, builds up a set of possible predicted values $\hat{S}_{t+h}^{(p,\alpha)}$ associated with the pattern α. Namely, for a given position to be predicted $t+h$, one composes a vector from time series observations according to the pattern α: $C = \left(y_{t+h-k_{L-1}^{(\alpha)}}, y_{t+h-k_{L-1}^{(\alpha)}-k_{L-2}^{(\alpha)}},\ldots, y_{t+h-k_{L-1}^{(\alpha)}-k_{L-2}^{(\alpha)}-\ldots-k_1^{(\alpha)}} \right)$. All elements of the vector C are assumed to be known; they may be either observed or already predicted values. Then if the Euclidean distance between a vector C and a truncated motif $TruncC_\alpha$, $C_\alpha \in \Xi_\alpha$ (a truncated motif comprises all elements but the last one, $TruncC_\alpha = \left(\eta_1^{(\alpha)},\ldots,\eta_{L-2}^{(\alpha)}\right)$ for $C_\alpha = \left(\eta_1^{(\alpha)},\ldots,\eta_{L-1}^{(\alpha)}\right)$), then the last element of the motif C_α, $\eta_{L-1}^{(\alpha)}$, is a reasonable estimate for a value to be predicted.

Respectively, the set $\hat{S}_{t+h}^{(p,\alpha)}$ is determined as $\hat{S}_{t+h}^{(p,\alpha)} = \left\{\eta_{L-1}^{(\alpha)} : \rho(C, TruncC_\alpha) \le \varepsilon\right\}$, where ε is a small threshold. In turn, a set of possible predicted values for a position $t+h$ is a union of the sets $\hat{S}_{t+h}^{(p,\alpha)}$ over all possible patterns, $\hat{S}_{t+h}^{(p)} = \bigcup_{\alpha\in\beta\aleph} \hat{S}_{t+h}^{(p,\alpha)}$. The unified predicted value $\hat{y}_{t+h}$ is calculated as a function of the set of possible predicted values for this position $\hat{S}_{t+h}^{(p)}$.

Obviously, in order to obtain predicted values for a fairly large prediction horizon h, one should apply this prediction routine iteratively, using already predicted values at intermediate positions $\hat{y}_{t+i}$ as inputs for new runs of the routine. Each run is called a prediction operation; a prediction operation generalizes a step in the iterated strategy. It is worth stressing again that, in order to make a prediction for some position, it is not necessary to make predictions for all intermediate positions – one may step-stone, leaving some intermediate positions unpredicted.

A consistent application of this procedure produces an algorithm to construct an operator $f_h : \hat{S}_{t+h} = f_h(\{y_t,\ldots,y_{t-s+1}\})$, which yields a set of possible predicted values for a position $t+h$ to be predicted. Since the number of patterns is rather large, so the size of the set may be. This makes it possible to design various algorithms to determine a unified predicted value and identify non-predictable points. In what follows we discuss such algorithms in greater detail.

Besides a set of possible predicted values, one may concurrently build up a set of weights $\Omega_{t+h} = \left\{\omega_{t+h}^{(i)}\right\}$, $i = 1..N_h$, which characterize comparative

significance of possible predicted values $\hat{y}_{t+h}^{(i)}$. To determine weights, we utilize several alternatives, based upon the following ideas:

1. The first technique relies on the notion of the prediction length required to obtain a predicted value $\hat{y}_{t+h}^{(i)}$. Evidently, an algorithm introduces some error into a predicted value with each prediction operation. Hence, the more prediction operation is performed to obtain a predicted value, the less reliable the predicted value is. As the distances between the first and last positions for various patterns may differ essentially for large h, the number of prediction operations carried out to obtain various $\hat{y}_{t+h}^{(i)}$ may likewise essentially differ. To calculate this quantity, we use the following recurrence formula. One assigns unity weights $\omega_i = 1$ to observed values; furthermore, if a possible predicted value $\hat{y}_{t+h}^{(i)}$ is obtained with the employment of a pattern $\alpha = \left(k_1^{(\alpha)}, \ldots, k_{L-1}^{(\alpha)}\right)$ and the values $y_{t+h-k_{L-1}^{(\alpha)}}, y_{t+h-k_{L-1}^{(\alpha)}-k_{L-2}^{(\alpha)}}, \ldots, y_{t+h-k_{L-1}^{(\alpha)}-k_{L-2}^{(\alpha)}-\ldots-k_1^{(\alpha)}}$ (observed or predicted) used as inputs for this prediction operation have weights $\omega_{t+h-k_{L-1}^{(\alpha)}}, \omega_{t+h-k_{L-1}^{(\alpha)}-k_{L-2}^{(\alpha)}}, \ldots,$ $\omega_{t+h-k_{L-1}^{(\alpha)}-k_{L-2}^{(\alpha)}-\ldots-k_1^{(\alpha)}}$, then one calculates the weight of these value as an average of these weights times a step-down factor λ:

$$\omega_{t+h}^{(i)} = \frac{\lambda}{L-1} \sum_{l=1}^{L-1} \omega_{t+h-\sum_{j=1}^{l} k_j^{(\alpha)}} \tag{5}$$

To a unified predicted value, one assigns an average weight of all possible predicted values that take part in its calculation. Thus, the step-down factor λ ensures that values calculated with a larger number of prediction operations receive smaller weights. In the simulation, λ usually takes up a value of $\lambda = 0.99$ [*wl*] (weighted sum, length).

2. The second technique suggests that a weight ω_i is inversely proportional to the distance between observed values and the centre of the cluster chosen to make the prediction:

$$\omega_i = \frac{\varepsilon - \rho(C, TruncC_\alpha)}{\varepsilon} \tag{6}$$

, where ε is a small parameter used as a threshold for building up a set of possible predicted values. In the simulation, ε usually takes up a value $\varepsilon = 0.05$ (for a normalized series) [*wd*] (weighted sum, distance).

3. The third approach suggests that a weight ω_i is a product of weights of the first and second approaches [*w*] (weighted sum, combined).

2.4. A unified predicted value ($\hat{y}_{t+h} = g\left(\hat{S}_{t+h}^{(p)}\right)$ (1))

A basic dichotomy for algorithms to determine a unified prediction value is whether $p = 1$ or $p > 1$. If $p = 1$, a unified prediction value is determined using a set of possible predicted values associated with the current position, $\hat{y}_{t+h} = g\left(\hat{S}_{t+h}^{(p)}\right) \equiv g\left(\hat{S}_{t+h}\right)$ [*S*] (set). If $p > 1$, it also employs the sets associated with preceding positions [*T*] (trajectory).

For the case of $p = 1$, several techniques to determine a unified predicted value for a set of possible predicted values $\hat{S}_{t+h}$ are possible:

- Calculating an average value: $\hat{y}_{t+h} = \frac{1}{N_{t+h}} \sum_{i=1}^{N_{t+h}} \hat{y}_{t+h}^{(i)}$. [*av*] (an average)
- Calculating a weighted average value: $\hat{y}_{t+h} = \frac{1}{\sum_{j=1}^{N_{t+h}} \omega_j} \sum_{i=1}^{N_{t+h}} \omega_i \hat{y}_{t+h}^{(i)}$. [*wl*, *wd*, or *w*] (please, see above).
- Clustering a set $\hat{S}_{t+h}$ in order to select the largest cluster Q_{j^*}: $\hat{S}_{t+h} = \bigcup_j Q_j$, $Q_i \cap Q_j = \emptyset$, $q_j = \left|Q_j\right|$, $j^* = \underset{j}{argmax} q_j$. Two clustering algorithms, the modified Wishart [27] and DBSCAN [30] are employed. The latter gives a good account of itself for one-dimensional data, that is fully applicable to a set $\hat{S}_{t+h}$. Both algorithms do not require *a priori* information about the true number of clusters. A unified predicted value is calculated as $\hat{y}_{t+h} = \frac{1}{|Q_{j^*}|} \sum_{\hat{y}_{t+h}^{(i)} \in Q_{j^*}} \hat{y}_{t+h}^{(i)}$ [*cm*] (a maximum cluster).

4. This technique is similar to the previous one, but one clusters not all elements of $\hat{S}_{t+h}$, but only those with weights greater than a predefined threshold ω_0. Alternatively, one may discard υ percent of observations with the least weights, irrespective to weights absolute values [*cw*] (a maximum cluster, weighted).

5. The two last techniques can be readily extended to the fuzzy sets. Normalized weights could be viewed, quite naturally, as values of a membership function to belong to a set of possible predicted values. A clustering algorithm should be replaced by some algorithm to cluster fuzzy data [*fs*] (a fuzzy set).

6. Roulette wheel. Analogously to the previous techniques, one clusters here a set of possible predicted values, but chooses the cluster which centre will be used as a unified predicted value probabilistically. Probability for a cluster to be chosen is directly proportional to the number of elements it contains, or the normalized sum of weights associated with its elements [*rw*] (a roulette wheel).

7. The most frequent value. One divides a range of values for a series in question, into disjoint intervals of equal lengths ε: $Q_i \cap Q_j = \emptyset$, $q_j = \left|Q_j\right|$, $j^* = \underset{j}{argmax} q_j$. An average value calculated over the elements of the most

'populated' interval, $\hat{y}_{t+h} = \frac{1}{|Q_{j^*}|}\sum_{\hat{y}_{t+h}^{(i)} \in Q_{j^*}} \hat{y}_{t+h}^{(i)}$, serves here as a unified predicted value [*mf*] (the most frequent).

8. Randomly perturbed most frequent value. One follows the previous technique, but adds a normally distributed noise to the average value: $\hat{y}_{t+h} = \frac{1}{|Q_{j^*}|}\sum_{\hat{y}_{t+h}^{(i)} \in Q_{j^*}} \hat{y}_{t+h}^{(i)} + \zeta(\Delta)$, where $\zeta(\Delta)$ is a normally distributed random variable with zero mean and variance $\Delta \geq 0$ [*mp*] (the most frequent, perturbed).

For the case of $p > 1$, we utilize the concept of a possible predicted trajectory. A possible predicted trajectory is defined as a sequence of possible predicted values at positions (not necessarily successive) preceding a prediction horizon and at the prediction horizon itself. In the framework of this approach, a prediction method produces not a sequence of sets of possible predicted values, but a set (a bunch) of possible trajectories that end in a prediction horizon. For some methods, the trajectory is built up by adding a random component at each position, and thus the maximum number of trajectories is a parameter of the respective prediction algorithm. For other methods, one chooses at each point, not a single unified prediction value, but several such values, thereby making a 'fan' of trajectories. Clearly, the number of trajectories grows exponentially, and to avoid combinatorial explosion, one should introduce constraints on this quantity, as if placing trajectories in a kind of 'pipe'.

$\hat{\xi}_{t+h}^{(s)} = \left(\hat{y}_{t+1}^{(s)}, \ldots, \hat{y}_{t+h}^{(s)}\right)$ denotes a s th possible predicted trajectory; $\hat{\xi}_{t+h}^{(s)}(i) = \hat{y}_{t+i}^{(s)}$, its value at an ith position; S_{max}, the maximum number of trajectories. $\hat{\Xi}_{t+h}$ denotes a set of all trajectories that end at a position $t+h$; $\hat{\Xi}_{t+h}(i)$, a set of their values at an ith position. Quite naturally, $\hat{S}_{t+h} \equiv \hat{\Xi}_{t+h}(h)$. It is worthy to note that since a trajectory may not contain all successive points, generally a set $\hat{\Xi}_{t+h}$ does not coincide with a set $\hat{S}_{t+h}^{(p)}$.

In the case of $p > 1$, some techniques to build up sets of possible predicted trajectories and then determine a unified predicted value are possible:

1. Trajectories with random perturbation. This technique employs a conventional iterative strategy (predicted values are calculated step by step, at all intermediate positions). A unified predicted value is calculated for each trajectory independently, with the employment of its own sets of possible predicted values. This value is equal to a perturbed average value over elements of the largest cluster (cf. the technique № 8 of the previous list): $\hat{y}_{t+h} = \frac{1}{|Q_{j^*}|}\sum_{\hat{y}_{t+h}^{(i)} \in Q_{j^*}} \hat{y}_{t+h}^{(i)} + \varsigma(\Delta)$, where $\varsigma(\Delta)$ is a normally distributed random variable with zero mean and variance $\Delta \geq 0$. In general, this technique closely resembles the respective technique for $p = 1$, but it requires that the procedure to build a possible

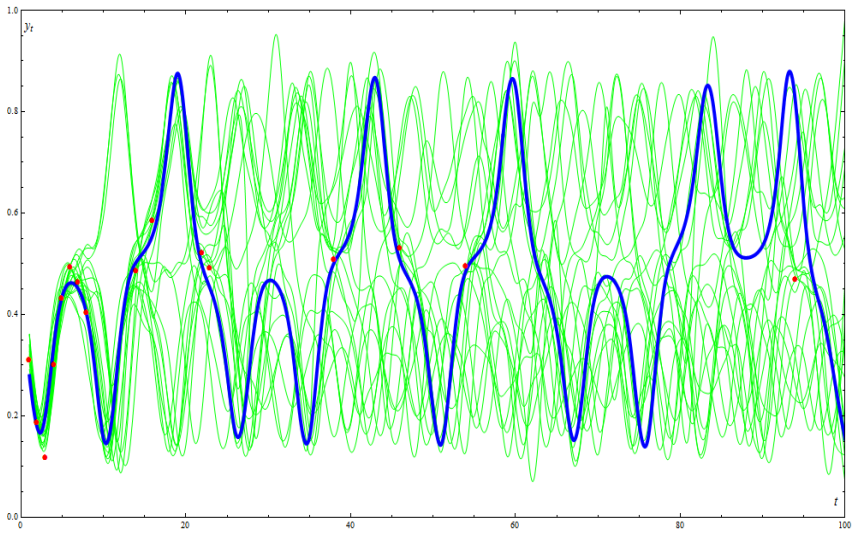


Fig. 3. A bunch of possible predicted trajectories for the Lorenz series. Blue thick solid line stands for a true trajectory; thin green lines, for possible trajectories; red points, for predictions made by the algorithm

predicted trajectory be repeated S_{max} times. Figure 3 shows a bunch of possible predicted trajectories for the Lorenz series [*tp*] (a trajectory, perturbed).

2. Trajectories that rely on a choice of several clusters (the garden of forking trajectories, a fan of trajectories). Similarly to the second technique of the previous list, one clusters (at each intermediate position) a set of possible predicted values and considers the following centres of the clusters: $\hat{S}_{t+k} = \bigcup_j Q_j,\ Q_i \cap Q_j = \emptyset$. However, in that case one chooses several centres, thus making trajectories to branch out. Usually, the first centre is that of the largest cluster, other ones are those of the second, third largest, and\or the centres of randomly chosen clusters. If all clusters are chosen randomly, one chooses among clusters that are larger than a certain predefined threshold, q_{min}, only. Irrespective of the way the centres are chosen, each one becomes the next element of a separate trajectory. In order to cut the number of the trajectories employed, we introduce constraints based on their weights. A weight of a trajectory is defined as a product of the weights of its elements; weights may be calculated by any of the three ways discussed above. To elucidate, a fan of trajectories is transformed into a pipe. For the cutting procedure, the current length (the number of elements) of trajectories should exceed a certain predefined threshold. As above, one may exclude trajectories with a weight less than a certain value ω_0 or υ percent of 'lightest' trajectories, regardless of however large their weights are [*tp*] (a trajectory, multiple clustering).

3. Trajectories that rely on reduced initial information. The technique combines, in a sense, iterated and direct strategies [9]. It suggests to build up not only a principal trajectory, starting at point t, but also a series of additional trajectories, starting at points $t-1, t-2,\ldots,t-a$, where a is a parameter. For an additional trajectory, observed values after its starting point are neglected and replaced by predicted values, calculated by the prediction algorithm in a usual way [*tr*] (a trajectory, reduced information).

For all techniques, a unified predicted value is calculated as an average over the last values of the above trajectories: $\hat{y}_{t+h} = \frac{1}{S_{max} \sum_{j=1}^{S_{max}\Sigma(h)} \hat{\xi}_{t+h}^{(j)}}$. It is worth noting that the second technique appears to be the most efficient, at the sacrifice, quite naturally, of computational efficiency.

Interestingly, for the predictive-clustering algorithms discussed in this study, a set of non-predictable points appears to closely approximate a set of points for which the same algorithm, without a unit that identifies non-predictable points, fails to make at least a satisfactory prediction. This observation provides the basis for a strategy to design various non-predictable-points identifying techniques.

2.5. Quality measures for algorithms to identify non-predictable points

The prediction method under discussion employs, most advantageously, various non-predictable-points identifying techniques. Such techniques use, as input data, either a set of possible predicted values for this position $\hat{S}_{t+h}^{(p)}$ $(p = 1)$ or a set of possible predicted trajectories that end at this point $\hat{\Xi}_{t+h}$ $(p > 1)$. For each prediction horizon h one may consider a set $NP(t, m, h) = \left\{t + i : t + m \leq t + i \leq t + h \ \& y_{t+i} \in Y_2 \& \zeta\left(\hat{S}_{t+i}^{(p)}\right) = 0\right\}$; then its set of non-predictable intermediate position is defined as $NIP(t, h) = NP(t, 1, h)$. A set of all non-predictable points of a test set is defined as $NP(h) = \bigcup_{t+h: y_{t+h} \in Y_2} NP(t, h, h) \equiv \left\{t + h: \ y_{t+h} \in Y_2 \& \zeta\left(\hat{S}_{t+h}^{(p)}\right) = 0\right\}$. This set consists of all positions of a test set such that the prediction algorithm fails to predict, when it predicts h steps ahead. Importantly, a point may belong to the set of non-predictable points if: (1) its set of possible predicted values is empty, or (2) this set comprises predicted values such that it is impossible to obtain an adequate prediction based on them. The functional (1) in effect minimizes the size of this set.

Such sets depend heavily on an algorithm used for a one-step-ahead prediction, f_h, a value of p, and a technique used to identify non-predictable points ζ. To design baselines for non-predictable-points identifying algorithms – and to compare them (for a given f_h) – we consider two extreme cases. For the first one, we do not employ a non-predictable-points identifying algorithm altogether: $\zeta(S) \equiv 1, NP(h) \equiv \emptyset, \ \forall h \in N$ by default. It means the prediction algorithm uses the closest motif regardless of how remote it is [*fp*] (forced prediction). A kind of light version of this baseline does not use motifs that are farther than a certain threshold [*ab*]. This version (the first extreme case) serves as a baseline for other.

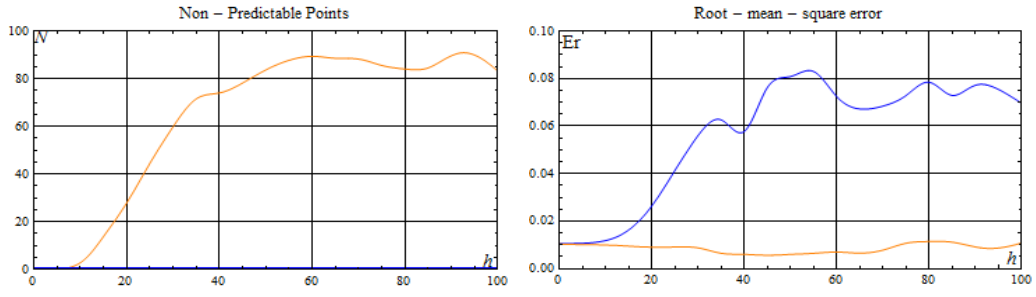
For the second extreme case, we assume that the prediction algorithm possesses *a priori* information about observed values at intermediate positions.

More precisely, if the difference between its unified predicted value and the true value at the same position is more than ε (a small parameter), then the algorithm is notified about this fact, and one recognizes the point as non-predictable. One should stress that this algorithm *does not* replace its predictions by the true values; it just does not take clearly erroneous predictions into account during the next prediction operations. Such points comprise, for a given prediction algorithm, its set of the ground-truth non-predictable points. Namely, for each point to be predicted $t+h$ we can build up the set

$GTNP(t,m,h) = \{t+i: t+m \le t+i \le t+h \ \&y_{t+i} \in Y_2 \& \rho(y_{t+i}, \hat{y}_{t+i}) \ge \varepsilon\}$, where $\hat{y}_{t+i} = g\left(\hat{S}_{t+i}^{(p)}\right)$ is, as usual, a unified predicted value calculated for a set of possible predicted values $\hat{S}_{t+i}^{(p)}$. Hereinafter, the distance is Euclidean. Then, similarly to the above, its set of the ground-truth non-predictable intermediate points is defined as $GTNIP(t,h) = GTNP(t,1,h) = \{t+i: t+1 \le t+i \le t+h \ \&y_{t+i} \in Y_2 \& \rho(y_{t+i}, \hat{y}_{t+i}) \ge \varepsilon\}$ and the set of the ground-truth non-predictable points for a test set, as $GTNP(h) = \bigcup_{t+h: y_{t+h} \in Y_2} GTNP(t,h,h) \equiv \{t+h: \ y_{t+h} \in Y_2 \& \rho(y_{t+i}, \hat{y}_{t+i}) \ge \varepsilon\}$.

If one, attempting to predict at $t+h$, excludes all intermediate points that fall into the set $GTNIP(t,h)$, one gets the second extreme case, that is, the prediction method that employs non-predictable points identifying algorithm using *a priori* information. (In our internal team slang, this algorithm is named 'the daemon' – Socrates is known to have his own daemon, who gave him advice in his most painful situations.) It is worthy to note that wide-raging simulation reveals that the results of such a prediction algorithm is nearly independent on the threshold value ε.

Quite naturally, there is no free lunch for real-world algorithms: they do not possess information about the true values for intermediate positions y_{t+i}, and the sole pieces of information they have at their disposal are sets of possible predicted values $\hat{S}_{t+i}^{(p)}$, or sets of possible predicted trajectories $\hat{\Xi}_{t+i}$. Nonetheless, this algorithm is extremely useful in determining the low error boundary that any other prediction algorithm

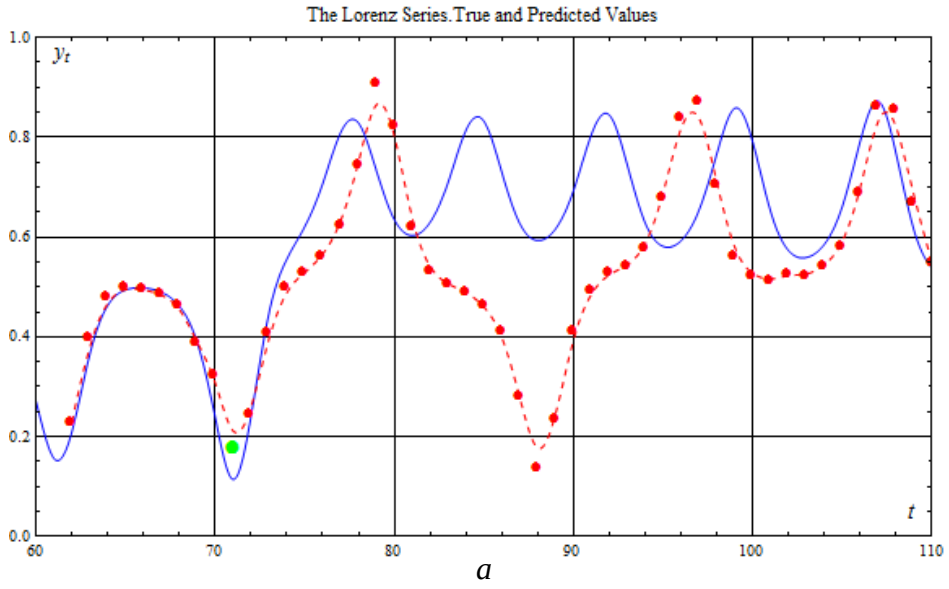


a b

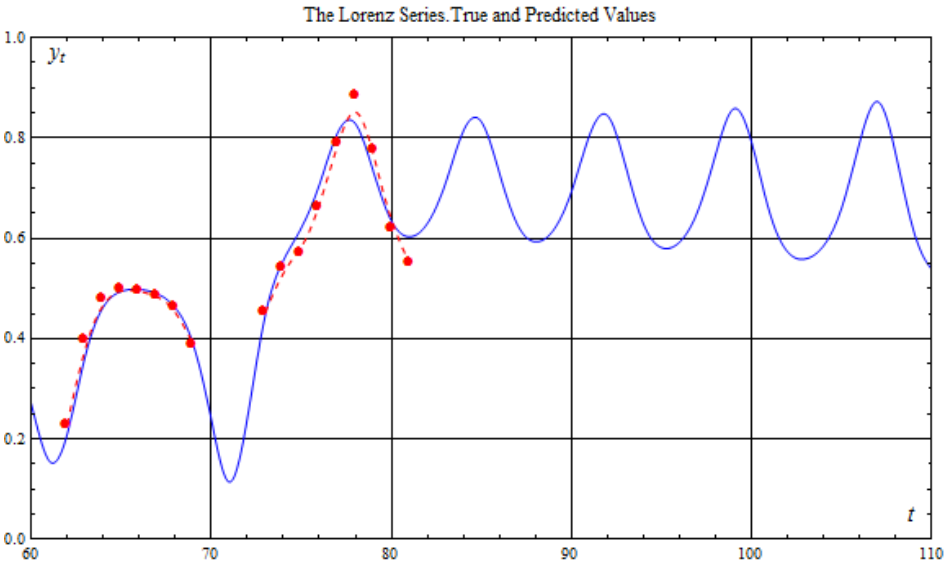
Fig. 4. The number of non-predictable points (a) and RMSE (b) vs. a prediction horizon for the two extreme cases. The blue curves correspond to the first extreme case (intermediate non-predictable points are not identified at all); orange, to the second one (intermediate non-predictable points are identified with *a priori* information)

may reach. One should try to develop non-predictable-points identifying techniques in such a way that their results will be as close as possible to this low boundary.

Graphs in Fig. 4 exemplify dependences of the number of non-predictable points (Fig. 4 *a*) and an average error on predictable points (Fig. 4 *b*) on the prediction horizon h for the two extreme cases. The blue curves correspond to the first extreme case (intermediate non-predictable points are not identified at all), and orange, to the second (intermediate non-predictable points are identified with *a priori* information).



a



b

Fig. 5. True (blue solid) and intermediate predicted (red dashed) values for the Lorenz series for the first (a) and second (b) extreme cases. A green disk indicates a point that the algorithm that uses *a priori* information identifies as non-predictable

From this figure we notice that in the first case, the number of non-predictable points naturally is equal to zero and a prediction error grows exponentially with h. The second extreme algorithm demonstrates quite the opposite results: the number of non-predictable points grows exponentially with h, while the prediction error function is nearly constant, bounded, and rather small. It is obvious that the first algorithm minimizes the functional (3), neglecting the functional (4); the second one, vice versa, minimizes the functional

(4), neglecting the functional (3). Figures 5a and 5b display the true values (blue solid) and intermediate predicted values (red dashed) for the Lorenz series for the first (5 a) and second (5 b) extreme cases. From the latter subfigure we notice that the second algorithm does not make a prediction for all points, but if it decides to predict, it provides rather accurate predictions. On the other hand, we notice that in the first case the predicted trajectory diverges from the true one just after the point (a green disk in Fig. 5 a) that is identified as non-predictable by the second algorithm – the first algorithm must predict at this point, as it must do at any point, and this immediately ruins the MSA prediction process, causing the predicted trajectory to diverge from the true one.

Any other prediction algorithm is a trade-off between the two extreme algorithms. In the context of multi-step ahead prediction, the second case is essentially more interesting, since it allows one to predict a fairly large number of steps ahead. Consequently, all algorithms discussed below attempt to follow suit with this very algorithm. (In the internal team's slang, to design such algorithms was referred to as 'to approximate the daemon'. :))

Common sense (and large-scale simulation) suggests that such an algorithm will work efficiently, if and only if the set of identified non-predictable points $NP(h)$ will be sufficiently close to the set of the ground-truth non-predictable points $GTNP(h)$. Symmetric difference of the two sets constitutes an auxiliary quality measure for a non-predictable-points identifying technique:

$$I_3 = |GTNP(h) \ \Delta \ NP(h)| \qquad 7)$$

It is possible to observe two contrary situations: (1) the difference $|GTNP(h) \setminus NP(h)|$ is large, whereas the difference $|NP(h) \setminus GTNP(h)|$ is small; (2) the difference $|NP(h) \setminus GTNP(h)|$ is large, whereas the difference $|GTNP(h) \setminus NP(h)|$ is small. The first case seems to correlate with an overly optimistic algorithm, the second, with an overly conservative.

2.6. Algorithms to identify non-predictable points (function ζ (1))

These algorithms show a basic dichotomy between those that employ a set of possible predicted values $\hat{S}_{t+h}$ $(p = 1)$ and those that employ a set of possible predicted trajectories $\hat{\Xi}_{t+h}$ $(p > 1)$. In order to gain a better insight into identification algorithms, one should consider a 'paragon' of a predictable point. We believe that its set of possible predicted values should consist of a single compact cluster, which includes an overwhelming majority of elements, and a number of small clusters and/or individual elements. On the contrary, non-predictable points are associated either with a continuum of possible predictable values, not amenable to be clustered, or, vice versa, with several approximately

equinumerous compact clusters. In terms of probability distributions, a predictable point correlates with a unimodal symmetric distribution with relatively small variance, a non-predictable one, with a nearly uniform distribution or with a distribution with several pronounced modes.

In order to characterize non-predictable points, we employ the following quantities: multimodality, defined as a percent of possible predicted values remote from the main mode; a difference between α and $1-\alpha$ percentiles; the second, third, fourth central moments, and the kurtosis of the distribution; its entropy as well. For each feature separately, we performed large-scale simulation on the validation set Y_3 to build up, sets of non-predictable points and the ground-truth non-predictable points ($NP(h)$ and $GTNP(h)$). A comparison of these sets makes it possible to estimate the best threshold values for each feature. It also reveals that each feature separately, except entropy, do not lead to large values of the objective (7) [*en*] (entropy).

In terms of clustering of a set of possible predicted values, we employ the following quantities: the number of clusters; the percent of possible predicted values that fall into the largest cluster; and the difference between percent of the values that belong to the largest and smallest clusters. Similarly to the previous case, it is revealed that these quantities also do not 'work' separately. Consequently, the main course to follow is to use them together, and together with quantities based on a distribution of possible predicted values.

We employ the following methods to separate a space of the features mentioned above into regions corresponding to predictable and non-predictable points: (1) logistic regression [*lr*]; (2) support vector machine [*sv*]; (3) decision tree C 4.5 [*dt*]; (4) k-nearest neighbours clustering [*kn*]; (5) multilayer perceptron with a single layer and 8 neurons in it [*mp*]. Mathematically, to apply this approach one should introduce, apart from a training set (Y_1) and a test set (Y_2), a third validation set, Y_3, disjoint with two others: $Y_1 \cap Y_3 = \emptyset$, $Y_2 \cap Y_3 = \emptyset$. To train these models, we label the validation set using the ground-truth non-predictable points set. Since for large prediction horizons the number of non-predictable points is essentially larger than that of predictable points, a sample is balanced by SMOTE methods (synthetic minority over-sampling technique) [31]; the method generates new elements in the neighbourhood of the smaller cluster. In addition to the above classifying methods, we employ their ensembles: (1) AdaBoost (Adaptive Boosting), the algorithm implies that each subsequent classifier is trained on elements wrongly classified by a current classifier [*al* or *as*] (adaptive boosting for logistic regression or support vector machine, respectively); (2) Stacking, the algorithm implies that several classifiers are applied to produce a feature space for a combiner algorithm (logistic regression) [*sl*] (stacking, logistic regression); (3) Voting, the algorithm assigns a label to the element by voting of classifiers [*vl*] (voting, logistic regression).

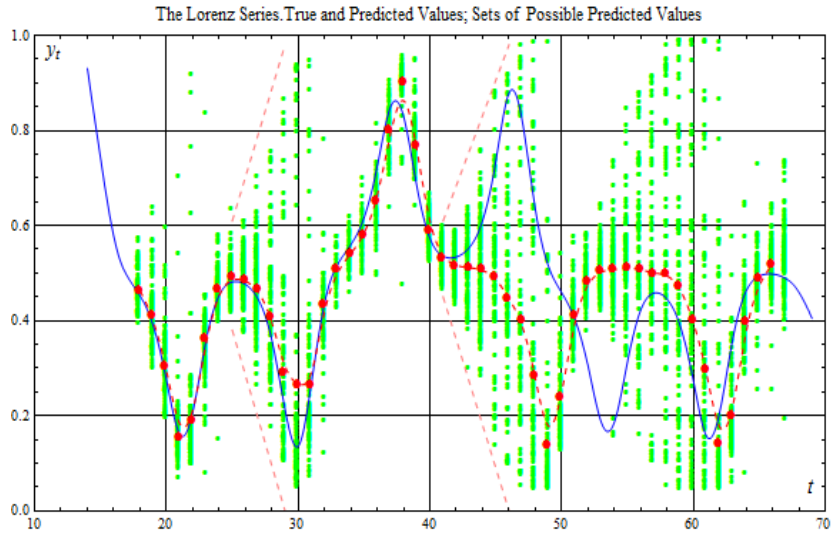


Fig. 6. Divergence of true (solid blue lines) and predicted (dashed red lines) trajectories correlates with a monotonic spread growth (indicated by pink dashed lines) of sets of possible predicted values (Each set is green disks with the same x-coordinate)

For the second approach, when one analyses a set of possible predicted trajectories $\hat{\Xi}_{t+h}$, trajectories convergence at a certain position is indicative of predictability. By the way of illustration, plots in Fig. 5 exemplify (for the Lorenz series) a set of possible predicted trajectories (thin green lines); a solid blue line shows a true trajectory for comparison; red disks denote predictions made by an algorithm (all other points are identified as non-predictable). From this figure we notice that predictable points are separated by sections of non-predictable points. On the other hand, predictable points are associated with regions where possible predicted trajectories converge with each other. It is worth noting that predicted values are in fairly good agreement with the respective true ones at the points where algorithms identified as predictable: the results shown in the figure correspond to a conservative variant of the algorithm; the algorithm seeks to minimize average error over the predictable points (the objective I_2 (4)), and to a lesser degree the number of non-predictable points (objective I_1 (3)). Figure 6 displays a true trajectory, a predicted trajectory, and sets of possible predicted values at intermediate positions. The figure illustrates the fact that a large spread (larger than some average value) of a set of possible predicted points is suggestive of the divergence between true and predicted trajectories. On the other hand, at such points one may observe several clusters far removed from each other instead of a single compact one.

These observations were utilized to develop several techniques to identify non-predictable points:

1. One applies the prediction algorithm to a validation set Y_3 in order to calculate an average spread over sets of possible predicted values $\kappa_{avg} = \frac{1}{|Y_3|}\sum_{t+h\in Y_3}\left|\widehat{max}\,\Xi_{t+h}(h) - \widehat{min}\,\Xi_{t+h}(h)\right|$. Then the algorithm recognizes a point as non-predictable, if its spread exceeds this average value times some factor $\vartheta\kappa_{avg}$ [*ls*] (large spread).

2. Unlike the previous case, one considers not the spread itself $\kappa(t+i) = \left|\widehat{max}\,\Xi_{t+h}(i) - \widehat{min}\,\Xi_{t+h}(i)\right|$, but its variation over some successive positions i. If it increases monotonically, then the first point is classed as non-predictable. Simulation suggests that the best option here is to consider three successive points. One may notice (please, refer to Fig. 6) that the divergence of true and predicted trajectories correlates with a monotonic spread growth (indicated by pink dashed lines) [*rg*] (rapid growth). One may also follow growth of the number of possible predicted value clusters obtained with DBSCAN [*rd*] or the Wishart clustering [*rw*].

3. For the trajectories that rely on reduced initial information, one may compare the last value of a trajectory starting at a position t, $\hat{y}_{t+h} = \hat{\xi}^{(0)}_{t+h}(h)$, with the weighted average of the last values for trajectories starting at positions $t-1$, $t-2$,…, $t-a$: $\tilde{y}_{t+h} = \sum_{j=1}^{a} \omega_j \hat{\xi}^{(j)}_{t+h+j}(h)$. If $|\hat{y}_{t+h} - \tilde{y}_{t+h}|$ is more than small ε, then the point is recognized as non-predictable. The weights are $\omega_j = \frac{1}{2^j}$; hence trajectories starting from positions closer to t possess larger weights [*ca*] (compare with average).

4. Another variant, designed for the trajectories that rely on reduced initial information, implies that one compares an averaged modulus of difference between the last point of a trajectory starting at a position t and a trajectory starting at other positions $\frac{1}{a}\sum_{j=1}^{a}\left|\hat{\xi}^{(0)}_{t+h+j}(h) - \hat{\xi}^{(j)}_{t+h+j}(h)\right|$ with ε [*ad*] (averaged difference).

5. Similarly to the previous case, one calculates a weighted average of the differences $\sum_{j=1}^{a}\omega_j\left|\hat{\xi}^{(0)}_{t+h+j}(h) - \hat{\xi}^{(j)}_{t+h+j}(h)\right|$ with weights equal to $\omega_j = \frac{2(a-j+1)}{a(a+1)}$ [*wa*] (weighted average).

6. It is also possible to apply all techniques from the previous list to a set of final points of the trajectories $\hat{\Xi}_{t+h}(h)$ [*wd*] (weighted, difference).

For comparison, tables of the next section include results for the extreme cases. Namely, a dash symbolizes that non-predictable points are not identified altogether, predicted values are forcibly calculated at all intermediate points; *ab* (absent) implies that a set of non-predictable points consists only of the points that the algorithm failed to find motifs close enough ($\hat{S}^{(p)}_{t+h} = \emptyset$), sets of possible predicted values are not analysed (the first extreme case); *id* (ideal), sets of possible predicted values are constructed using *a priori* information (the second extreme case).

3. Numerical results

The method discussed in the previous section was applied to a time series generated by the Lorenz system and hourly electricity load series for Germany. The aforementioned clustering algorithm is applied to generate samples

employing all possible patterns of four elements ($L = 4$) with the maximum distance between neighbouring positions in the pattern equal to K_{max}10. So the number of patterns used amounts to $|\aleph(L, K_{max}())|||$=1000. If the algorithm employs not all possible patterns, the percentage of used patterns is equal to 4 %, $|\beta\aleph(L, K_{max}())|||$=40.

For each series, in order to display results graphically, we utilize two types of plots. The first type (please, refer to Appendix A) shows a predicted trajectory up to a prediction horizon (a dashed red line with disks of predicted values); a true trajectory is shown for comparison (a solid blue line). For the predicted trajectory, the first presented value corresponds to a one-step ahead prediction, the second, to a two-step ahead prediction, and so on. Plots of the first type are intuitively obvious, but seem to be less instructive than those of the second type. Plots of this type display an error measure (the number of non-predictable points, the root-mean-square error [RMSE], the mean absolute percentage error [MAPE]) against a prediction horizon. Each such figure also contains, for comparison, the respective plots for two extreme cases (please, refer to Section *Quality measures for algorithms to identify non-predictable points*): for the algorithm that does not identify non-predictable points (blue) and the one that identifies them using *a priori* information (orange, or red, if only 4% of patterns is employed). The two plots make a 'fork' that usually bounds from below and above plots of all other discussed algorithms.

These results are presented in tabular form for several typical *prediction horizons* h. We used two types of tables. The first four columns present characteristics of the method used and are the same for both types. Namely, the first column indicates a percentage of employed patterns β and the way the algorithm obtains motifs (please, refer to the Section *A training set*). The second column indicates whether the algorithm builds up a set of possible predicted values or predicted trajectories, and how it calculates a unified predicted value (please, refer to the Sections *Sets of possible predicted values* and *A unified predicted value*). The third column provides information on the way the algorithm identifies non-predictable points (please, refer to the Section *Algorithms to identify non-predictable points*) with the following possible values. For information on symbolic labels used in these columns, please refer to the previous section (in square brackets after the appropriate technique) or to tables in Appendix B. The fourth column indicates the figure that displays results obtained using the respective variant of the algorithm and the colour used to plot

the respective curves. Finally, all other columns present error measures for prediction horizons $h = 1$, 10, 50 and 100 (the number of non-predictable points, the mean absolute percentage error, and the root-mean-square error). It is worth noting that the first ten lines correspond to the extreme cases (please, refer to the Section *Quality measures for algorithms to identify non-predictable points*).

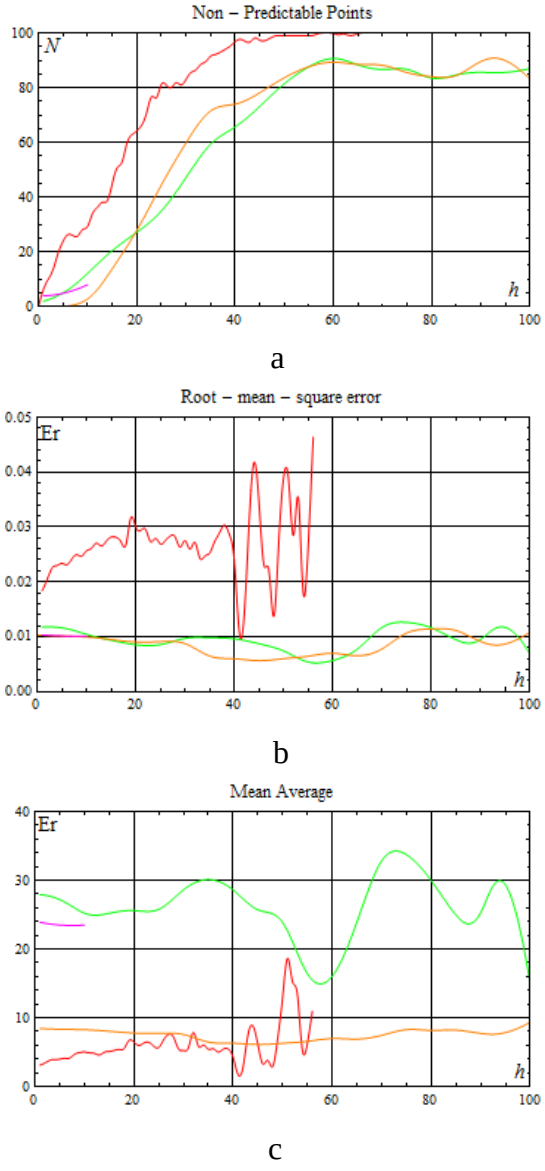


Fig. 7. The second extreme case. Results for various versions of algorithms that identify non-predictable points with *a priori* information. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon Green and red curves correspond to the method that employ the Wishart clustering (100% and 4%, respectively); orange and magenta, to those that does not employ clustering algorithms (100% and 4%, respectively).

Tables of the second type present information on sets of non-predictable points for various variants of the method considered. The first columns coincide with the respective columns of the first type tables. Other columns indicate, for the same prediction horizons ($h = 1$, 10, 50 and 100), $Re\,call = \frac{TP}{TP+FN}$, $Pr\,ecision = \frac{TP}{TP+FP}$, and F -measure, that is the harmonic mean of two previous quantities: $F = \frac{2\,Pr\,ecision * Re\,call}{Pr\,ecision + Re\,call}$.

The Lorenz system [4, 32] with standard 'chaotic' parameters $\sigma = 10, b = \frac{8}{3}, r = 28$, integrated with the employment of the Runge-Kutta fourth-order method (integration step is equal to $\Delta t = 0.1$), yields a time series hereinafter referred to as the Lorenz series. The series in question is a typical chaotic series; it is used as a conventional benchmark to test forecasting procedures for chaotic time series.

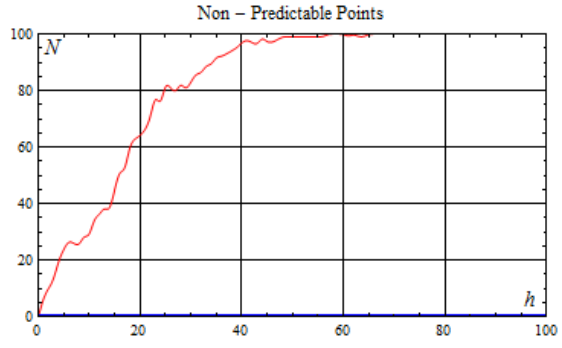
To distinguish chaotic and noisy time series, Rosso et al. [33] proposed to check the position of a pair of quantities, entropy and complexity, on the appropriate plane. They calculated these quantities for the Lorenz series: 0.68 and 0.45; this makes it possible to classify this series as chaotic. On the other hand, the highest Lyapunov exponent λ for this series, calculated using the Eckman algorithm [34], is equal to 0.92, which is in a good agreement with

results by Malinetskii and Potapov [4] (see on p. 217). A strictly positive value of this quantity also argues for series chaoticity.

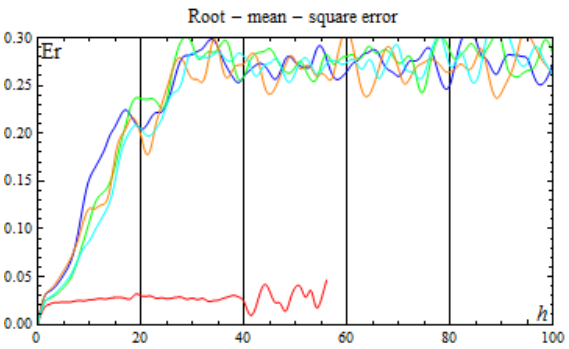
Numerical error for the fourth-order Runge-Kutta method amounts to $\varepsilon(0) = \Delta t^4 = 10^{-4}$. If the maximum prediction error is chosen as $10^{-1} = \varepsilon(T)(0)^{\lambda T}{}_{max}$, then it is straightforward to estimate a horizon of predictability as $T \sim \frac{1}{\lambda} ln \frac{\varepsilon_{max}}{\varepsilon(0)}$=7.5, or (in integration steps) 75 time series observations.

For the Lorenz series, the first 3000 observations are discarded in order to ensure that trajectory moves in the neighbourhood of the respective strange attractor. The test set for the series consists of 1000 observations; the training set consists of 10000. Table 1 and 2 are the first and second type tables for the Lorenz series.

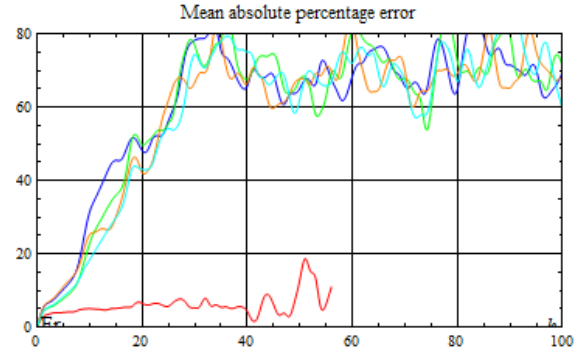
Large-scale simulation reveals that for the first extreme case (a point is recognized as non-predictable only if there are no motifs close enough to make prediction, $\hat{S}_{t+h}^{(p)} = \emptyset$) both the number of non-predictable points and average errors for predictable ones grow exponentially with a prediction horizon. Figure 7 exhibits the number of non-predictable points (7a), RMSE (7b), and MAPE (7c) vs. a prediction horizon in order to compare results obtained for the methods utilizing 100% and 4% of patterns. Namely, green and red curves correspond to



a



b



c

Fig. 8. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon (4% of patterns are used). Values are calculated (without identifying non-predictable points) as average of possible predicted values (blue) or a weighted average with weights inversely proportional to a distance to a cluster centre, weights proportional to predictive length, and combined weights (green, orange, and cyan lines, respectively). Red curves correspond to the algorithm that identifies non-predictable points with *a priori* information

the method that employ the Wishart clustering (100% and 4%, respectively); orange and magenta, to those that does not employ clustering algorithms (100% and 4%, respectively). Such results allow us to conclude that the more patterns are employed, the better the prediction quality: the percent of non-predictable points is nearly zero for a larger prediction horizon, whereas the error measures for predictable points are comparable.

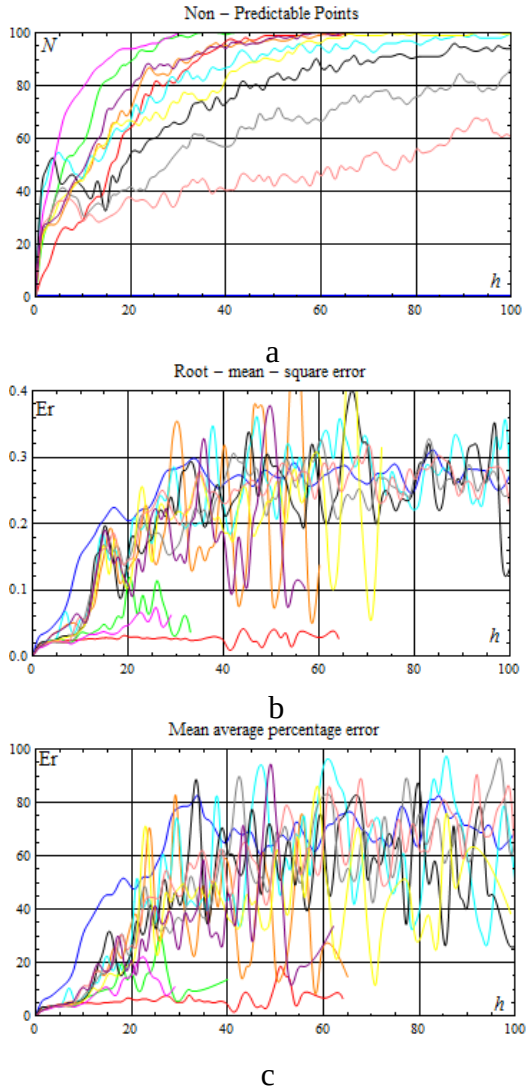


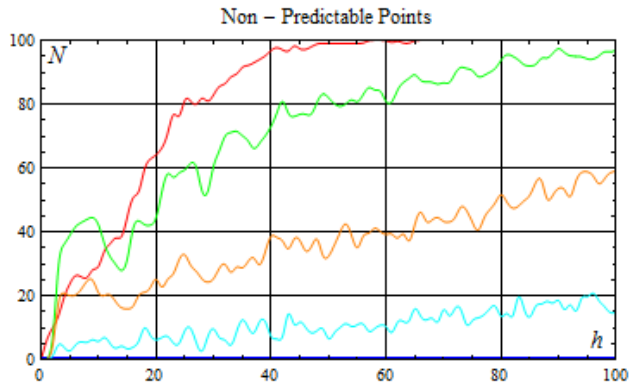
Fig. 9. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon (4% of patterns are used). The techniques identify non-predictable points with logreg clasification [*lr*] (green); support vector machine [*sv*] (orange); decision tree [*dt*] (cyan); knn [*kn*] (black); multilayer perceptron [*mp*] (gray); adaboost logreg [*al*] (magenta); adaboost svm [*as*] (yellow); voting [*vl*] (pink); stacking [*sl*] (orange). Red curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points.

Figure 8 presents prediction results for multi-step ahead prediction (the number of non-predictable points (8a), RMSE (8b), and MAPE (8c)); all methods does not identify non-predictable points altogether (4% of patterns is used). A unified predicted value is determined as an average (blue lines) [*av*] or a weighted average with weights inversely proportional to a distance to a cluster centre [*wd*], weights proportional to predictive length [*wl*], and combined weights [*w*] (green, orange, and cyan lines, respectively). For comparison, red lines represent the second extreme case, when the non-predictable points are identified with the employment of *a priori* information (please, refer to the Subsection *Quality measures for algorithms to identify non-predictable points*). One may conclude that the way the algorithm calculates a unified prediction value scarcely affects the rate of this growth. Consequently, of crucial importance is the algorithm used to identify non-predictable points. In what follows, we discuss results for various identification techniques.

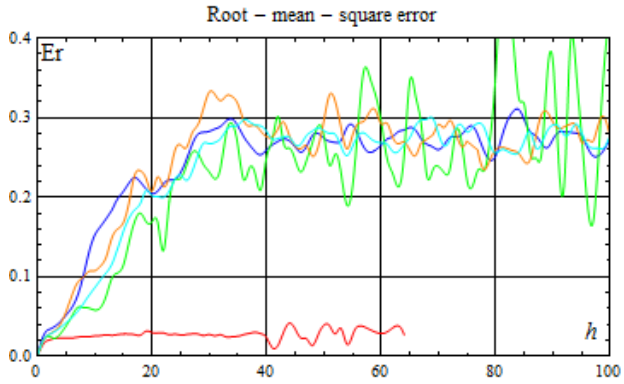
The first one is that that employs entropy of possible predicted values distribution. Simulation reveals that the optimal threshold between the entropies of predictable and non-predictable is equal to $\Delta = 0.1$. Figure A1 (please, refer to Appendix A) portrays a typical empirical distribution for entropy for the ground-

truth non-predictable points $GTNP(50)$ and its complement to a testing set $Y_2 \backslash GTNP(50)$. It is quite obvious that entropy values that correlate with the ground-truth non-predictable points and those that correlate with predictable, differ essentially. Unfortunately, it appears to be impossible to design an efficient test based solely on entropy values.

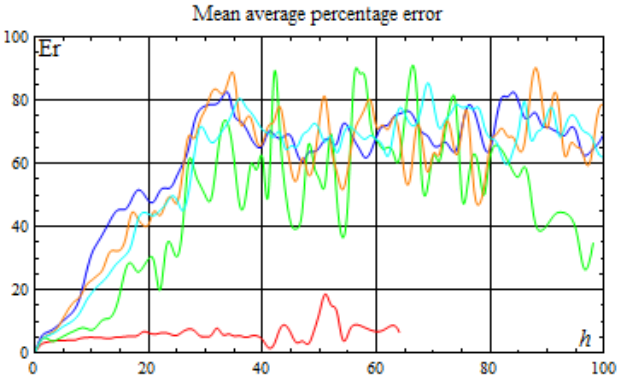
Figure 9 shows results obtained with various machine learning algorithms applied to a feature space build for sets of possible predicted values. The following parameters for the machine learning algorithms: multi-layer perceptron contains 8 hidden layers and utilizes sigmoidal activation function; support vector machine employs polynomial kernel functions; C4.5 employs entropy criterion without any restriction on tree depth; k-nearest neighbour value is equal to 3. Here, red curves correspond to non-predictable points identified with *a priori* information; blue, not identified at all; green, identified with logreg classifier [*lr*]; orange, with SVM classifier [*sv*]; cyan, with decision tree [*dt*]; black, with k-nearest neighbour [*kn*]; grey, with multi-layer perceptron classifier[*mp*]; magenta, with AdaBoost ensemble of logreg classifiers [*al*]; yellow, with AdaBoost ensemble of SVM classifiers [*as*]; pink, with voting of log-reg classifiers [*vl*]; orange, with stacking of log-reg classifiers [*sl*].



a



b

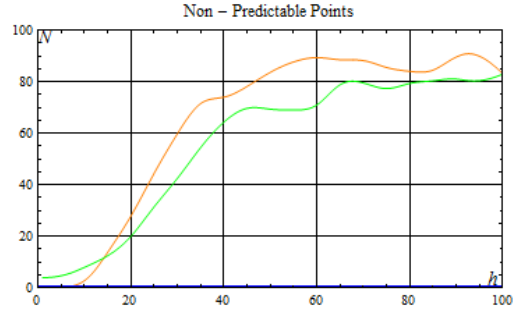


c

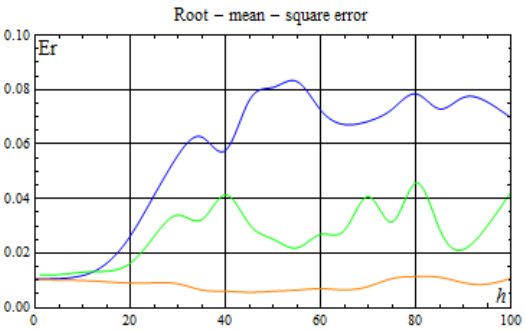
Fig. 10. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon (4% of patterns are used). The techniques identify non-predictable points based on growth of the spread [*rg*] (green); of the number of DBSCAN clusters [*sd*] (orange); of the number of Wishart clusters [*sw*] (cyan) over three consecutive points. Red curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points.

Among all machine learning methods considered, the one based on logistic regression [*lr*] (green curves) and Adaboost ensemble of them [*al*] (magenta curves) seem to stand out. They result in a nearly constant dependence of both error measures on non-predictable points on the horizon, and the error measures for predictable points are comparable with those for the algorithm that used *a priori* information to identify non-predictable points (the second extreme case, red curves). Unfortunately, it also demonstrates an exponential growth of the number of non-predictable points with a rate larger than that for the second extreme case. The method that employs a multi-layer perceptron [*mp*] (gray curve), vice versa, demonstrates a moderate rate of growth for the number of non-predictable points with rapid error growth. Figures A2-A5 exemplify predicted trajectories as compared with a true one for these non-predictable-points identifying techniques. All methods that use an ensemble of classifiers to distinguish predictable and non-predictable points show comparable efficiency. Figure 10 displays results for the algorithms that use growth of the spread [*rg*] (green); of the number of DBSCAN clusters [*sd*] (orange); of the number of Wishart clusters [*sw*] (cyan) over three consecutive points to identify non-predictable points. Unfortunately, these approaches do

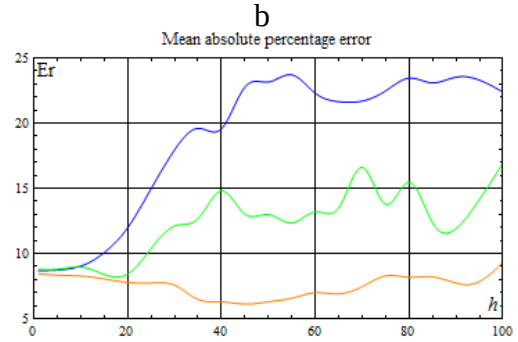
not demonstrate desirable behaviour. Figure A6 demonstrates true and predicted trajectories for this case.



a

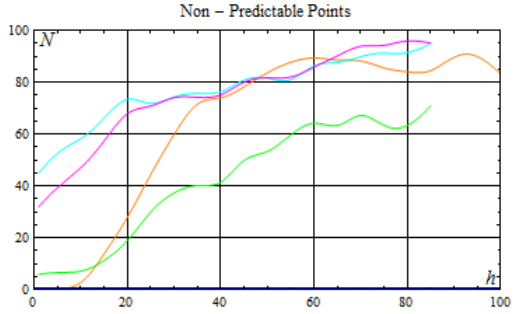


b

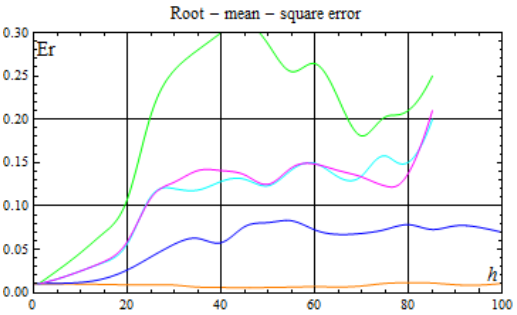


c

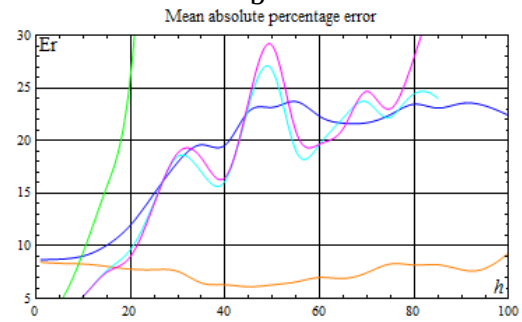
Fig. 11. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon for trajectories with random perturbation (100 % of patterns are used; no clustering technique is applied). The algorithm identifies non-predictable points when possible predicted trajectories diverge. Orange curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points.



a



b



c

Fig. 12. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon for trajectories with random perturbation (100 % of patterns are used; the Wishart clustering). The algorithm identifies non-predictable points if possible predicted trajectories diverge. Orange curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points.

In order to implement a branching trajectories approach (the concept of 'pipe'), we cluster points using DBSCAN algorithm and select those clusters that appear to be larger than q_{min}= 5% and 10% of the total number of possible predicted values. Of these clusters, we chose

two, using the following strategies: one may choose (1) the largest and the second largest clusters; (2) two randomly chosen clusters; and (3) the largest and randomly chosen ones. The maximum allowed number of trajectories (pipe diameter) is equal to 20. It was ascertained that exponential error growth for predictable points is peculiar to this technique likewise, though it is not as steep as for the previous case.

Simulation reveals that the methods that are based upon sets of predictable trajectories compare favourably with those based upon sets of predictable points. Figure 11 exhibits results for trajectories with random perturbation [*tp*]. The presented results correspond to a variance of the perturbation 0.05. In order to identify non-predictable points, the algorithm clusters a set of final points of the predicted trajectories. If a size of the largest cluster is less than 4% of the total number of the final points, then the algorithm recognizes this point as non-predictable. To cluster points, we use DBSCAN with the following parameters: eps = 0.01, min_samples = 5. (Orange curve corresponds to the method that does not identify non-predictable points; it differs strongly from the respective curves of the previous plots (the red ones), since the algorithm builds up sets of possible predictable values in different manner). Figure A8 presents typical true and predicted trajectories for this case.
An inspection of the figure indicates that the algorithm predicts values at positions $t + 100$, that significantly exceeds 75 (an estimated horizon of predictability for the Lorenz series, please, see above). This justifies the title of the present article. Figures A8 (please, refer to Appendix A) show typical curves of error measures against the number of possible trajectories (100 % of patterns are used; no clustering technique is applied). The algorithm identifies non-predictable points when possible predicted trajectories diverge. Orange curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points. $\Delta = 0.033$. Results for the methods that use reduced initial information are presented in Fig. 12.

Table 1. The Lorenz series. Non-predictable points and error measures

β %	PVs	NP	Figs	$h = 1$			$h = 10$			$h = 50$			$h = 100$		
motif	UPV	T		NP %	MA %	RMS	NP %	MA %	RMS	NP %	MA %	RMS	NP %	MA %	RMS
100 cl	*S av*	*ab*	-	0	8.7	0.011	0	8.8	0.013	0	22.5	0.08	0	22.4	0.07
100 cl	*S av*	*id*	7, green	2	8.9	0.011	12	8.3	0.01	82	7.2	0.007	82	9.5	0.01
100 p	*T av*	*ab*	4, blue	0	2.0	0.04	0	10	0.13	0	22	0.29	0	-	-
100 p	*T av*	*id*	7, orange	5	2.0	0.03	27	2	0.03	99	3	0.03	-	-	-
4 cl	*S av*	*ab*	-	0	5.7	0.03	0	32	0.154	0	63	0.281	0	69.7	0.28
4 cl	*S av*	*id*	7, red	7	3.1	0.018	28	5	0.025	99	7	0.029	100	-	-
4 p	*T av*	*id*	7, magenta	92	13.7	0.013	89	51.5	0.02	79	75.4	0.06	93	121	0.1
4 cl	*S wd*	*ab*	8, green	0	4.5	0.023	0	23.1	0.106	0	64.2	0.261	0	70.7	0.28
4 cl	*S wl*	*ab*	8, orange	0	5.7	0.03	0	25.9	0.124	0	67.2	0.285	0	68.6	0.26
4 cl	*S w*	*ab*	8, cyan	0	4.5	0.023	0	18.8	0.085	0	56.3	0.25	0	59.1	0.26
4 cl	*S av*	*en*	-	28	3.0	0.016	21	18.4	0.089	36	51.0	0.269	38	63.9	0.29
4 cl	*S w*	*lr*	9, green	18	2.2	0.014	57	6.3	0.038	100	-	-	100	-	-
4 cl	*S w*	*sv*	9, orange	24	2.1	0.014	46	6.0	0.039	97	52.1	0.326	100	-	-
4 cl	*S w*	*dt*	9, cyan	43	2.6	0.015	44	7.2	0.062	86	55.9	0.258	97	48.1	0.24

4 cl	*S w*	*kn*	9, black	44	4.0	0.019	42	6.0	0.042	84	73.7	0.319	**94**	**26.5**	**0.14**
4 cl	*S w*	*m*	9, grey	28	2.2	0.014	29	8.0	0.048	68	46.6	0.195	87	68.0	0.25
4 cl	*S w*	*al*	9, magenta	28	2.0	0.013	79	4.7	0.026	100	-	-	100	-	-
4 cl	*S w*	*as*	9, yellow	22	2.3	0.014	43	6.0	0.04	90	33.3	0.243	100	-	-
4 cl	*S w*	*vl*	9, pink	26	2.2	0.015	27	12.5	0.063	48	48.6	0.249	60	53.5	0.22
4 cl	*S w*	*sl*	9, orange	26	2.1	0.014	48	11.7	0.061	98	109.	0.371	100	-	-
4 cl	*S w*	*rg*	10, green	0	4.5	0.023	43	6.9	0.057	84	56.7	0.296	97	259.	0.45
4 cl	*S w*	*rd*	10, orange	0	4.5	0.023	20	23.0	0.105	31	58.6	0.260	59	77.7	0.27
4 cl	*S w*	*rw*	10, cyan	0	4.5	0.023	5	19.1	0.086	11	73.7	0.289	15	62.1	0.28
100 p	*T tp*	*rd*	11, green	0	8.3	0.012	0	9.0	0.028	84	13.0	0.042	**83**	**17.0**	**0.04**

The first column indicates a percentage of employed patterns β and the way the algorithm obtains motifs (please, refer to the Section *A training set*). The second column indicates whether the algorithm builds up a set of possible predicted values or predicted trajectories, and how it calculates a unified predicted value (please, refer to the Sections *Sets of possible predicted values* and *A unified predicted value*). The third column provides information on the way the algorithm identifies non-predictable points (please, refer to the Section *Algorithms to identify non-predictable points*) with the following possible values. For information on symbolic labels used in these columns, please refer to the previous section (in square brackets after the appropriate technique) or to tables in Appendix B. The fourth column indicates the figure that displays results obtained using the respective variant of the algorithm and the colour used to plot the respective curves. Finally, all other columns present error measures for prediction horizons $h = 1$, 10, 50 and 100 (the number of non-predictable points, the mean absolute percentage error, and the root-mean-square error). Bold green colour indicates the best results after a horizon of predictability obtained with sets and trajectories of possible predicted values.

Table 2. The Lorenz series. Sets of non-predictable points

β % motif	PVs UPV	NPT	Figs	$h = 1$			$h = 10$			$h = 50$			$h = 100$		
				Rcl	Prc	F	Rcl	Prc	F	Rcl	Prc	F	Rcl	Prc	F
100 cl	*S av*	*ab*	-	0.8	0.74	0.77	0.69	0.46	0.55	-	-	-	-	-	-
100 cl	*S av*	*id*	7, green	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
100 p	*T av*	*ab*	4, blue	1.0	0.73	0.84	1.0	0.73	0.84	0.03	1.0	0.05	-	-	-
100 p	*T av*	*id*	7, orange	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4 cl	*S av*	*ab*	-	1	0.93	0.96	1	0.72	0.84	1	0.01	0.02	-	-	-
4 cl	*S av*	*id*	7, red	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4 p	*T av*	*id*	7, magenta	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4 cl	*S wd*	*ab*	8, green	1	0.93	0.96	1	0.72	0.84	1	0.01	0.02	-	-	-
4 cl	*S wl*	*ab*	8, orange	1	0.93	0.96	1	0.72	0.84	1	0.01	0.02	-	-	-
4 cl	*S w*	*ab*	8, cyan	1	0.93	0.96	1	0.72	0.84	1	0.01	0.02	-	-	-
4 cl	*S av*	*en*	-	0.59	0.94	0.73	0.79	0.73	0.76	0	0	0	-	-	-
4 cl	*S w*	*lr*	9, green	0.78	0.99	0.87	0.28	0.83	0.42	-	-	-	-	-	-
4 cl	*S w*	*sv*	9, orange	0.71	1	0.83	0.57	0.77	0.66	0	0	0	-	-	-
4 cl	*S w*	*dt*	9, cyan	0.41	0.97	0.58	0.5	0.8	0.62	0	0	0	-	-	-
4 cl	*S w*	*kn*	9, black	0.42	0.98	0.59	0.61	0.78	0.69	0	0	0	-	-	-
4 cl	*S w*	*mp*	9, grey	0.7	0.97	0.81	0.89	0.77	0.83	1	0.01	0.03	-	-	-
4 cl	*S w*	*al*	9, magenta	0.72	1	0.84	0.2	0.94	0.34	-	-	-	-	-	-
4 cl	*S w*	*as*	9, yellow	0.77	0.97	0.86	0.6	0.83	0.69	0	0	0	-	-	-
4 cl	*S w*	*vl*	9, pink	0.73	0.97	0.83	0.85	0.77	0.81	1	0.01	0.03	-	-	-
4 cl	*S w*	*sl*	9, orange	0.71	1	0.83	0.57	0.77	0.66	0	0	0	-	-	-
4 cl	*S w*	*rg*	10, green	1	0.93	0.96	0.68	0.86	0.76	0	0	0	-	-	-
4 cl	*S w*	*rd*	10, orange	1	0.93	0.96	0.68	0.86	0.76	0	0	0	-	-	-
4 cl	*S w*	*rw*	10, cyan	1	0.93	0.96	0.93	0.71	0.8	1	0.01	0.02	-	-	-
100 p	*T tp*	*rd*	11, green	1.0	0.73	0.84	1.0	0.73	0.84	0.03	1.0	0.05	**0.05**	**0.8**	**0.1**

Legends of the first 4 columns are the same as in Table 1. Other columns indicate results for the prediction horizons of $h = 1$, 10, 50, 100, Recall, Precision, and F -measure.

The second series considered is a real-world time series (hourly load values in Germany, from 23:00 12/31/2014 to 14:00 20/02/2016 https://www.entsoe.eu/data/power-stats/). A pairing of entropy and complexity for this series amounts to (0.499, 0.372), that is a strong point in favour of its

chaoticity. Its highest Lyapunov exponent amounts to $0.125 > 0$, which supports the hypothesis that the series is chaotic.

All of the aforementioned methods were used in the analysis of this time series; results are demonstrated in tables 3 and 5, which are similar to tables 1 and 2 for the Lorenz series. For the sake of brevity, we present essentially lesser number of plots.

Table 3. The electricity load series. Non-predictable points and error measures

β %	PVs	NP	Figs	$h = 1$			$h = 10$			$h = 50$			$h = 100$		
motif	UPV	T		NP %	MA %	RMS	NP %	MA %	RMS	NP %	MA %	RMS	NP %	MA %	RMS
100 cl	*S av*	*ab*	-	0	14.0	0.007	0	22.3	0.019	0	35.6	0.077	0	42.1	0.083
100 cl	*S av*	*id*	7, green	22	10.4	0.005	32	11.4	0.004	64	12.0	0.005	57	9.2	0.006
100 p	*T av*	*ab*	4, blue	0	17.8	0.01	0	37	0.035	0	44.3	0.08	0	45.7	0.09
100 p	*T av*	*id*	7, orange	17	9.3	0.005	28	10.8	0.005	61	11.4	0.006	63	10.6	0.005
4 cl	*S av*	*ab*	-	0	4.9	0.031	0	31.0	0.154	0	48.4	0.268	0	38.7	0.272
4 cl	*S av*	*id*	7, red	8	3.7	0.022	66	4.6	0.027	100	-	-	100	-	-
4 p	*T av*	*id*	7, magenta												
4 cl	*S wd*	*ab*	8, green	0	3.9	0.025	0	23.0	0.120	0	43.0	0.249	0	33.4	0.241
4 cl	*S wl*	*ab*	8, orange	0	4.9	0.032	0	25.2	0.127	0	41.0	0.236	0	40.4	0.283
4 cl	*S w*	*ab*	8, cyan	0	3.9	0.025	0	19.0	0.102	0	39.8	0.233	0	39.4	0.277
4 cl	*S av*	*en*	-	47	2.3	0.016	31	18.6	0.117	48	43.0	0.270	72	39.5	0.304
4 cl	*S w*	*lr*	9, green	24	2.8	0.018	62	24.7	0.161	100	-	-	100	-	-
4 cl	*S w*	*sv*	9, orange	28	2.6	0.017	70	10.1	0.092	100	-	-	100	-	-
4 cl	*S w*	*dt*	9, cyan	61	2.0	0.016	91	4.8	0.025	100	-	-	100	-	-
4 cl	*S w*	*kn*	9, black	56	2.6	0.017	96	5.2	0.037	100	-	-	100	-	-
4 cl	*S w*	*mp*	9, grey	27	2.8	0.020	54	16.5	0.112	95	18.3	0.121	100	-	-
4 cl	*S w*	*al*	9, magenta	39	3.2	0.021	63	24.1	0.162	100	-	-	100	-	-
4 cl	*S w*	*as*	9, yellow	15	3.1	0.02	35	16.8	0.108	76	38.0	0.216	**92**	**34.3**	**0.322**
4 cl	*S w*	*vl*	9, pink	27	2.8	0.02	53	16	0.111	92	29.5	0.2	100	-	-
4 cl	*S w*	*sl*	9, orange	30	2.5	0.017	75	5.7	0.047	100	-	-	100	-	-
4 cl	*S w*	*rg*	10, green	0	3.9	0.025	11	19.3	0.117	55	43.0	0.244	73	40.4	0.315
4 cl	*S w*	*rd*	10, orange	0	3.9	0.025	9	21.4	0.115	47	45.2	0.228	59	42.9	0.317
4 cl	*S w*	*rw*	10, cyan	0	3.9	0.025	4	17.8	0.101	7	47.6	0.255	19	37.3	0.257
100 p	*T tp*	*rd*	11, green	0	10.3	0.005	0	11.7	0.006	43	34.1	0.05	**51**	**33**	**0.034**

Same legend as in Table 1.

Table 4. The electricity load series. Sets of non-predictable points

β %	PVs	NP	Figs	$h = 1$			$h = 10$			$h = 50$			$h = 100$		
motif	UPV	T		Rcl	Prc	F	Rcl	Prc	F	Rcl	Prc	F	Rcl	Prc	F
100 cl	*S av*	*ab*	-	1.0	0.78	0.88	1.0	0.68	0.81	1.0	0.36	0.53	1.0	0.43	0.6
100 cl	*S av*	*id*	7, green	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
100 p	*T av*	*ab*	4, blue	1	0.83	0.91	1	0.72	0.84	1	0.39	0.56	1	0.37	0.54
100 p	*T av*	*id*	7, orange	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4 cl	*S av*	*ab*	-	1	0.92	0.96	1	0.34	0.5	-	-	-	-	-	-
4 cl	*S av*	*id*	7, red	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4 p	*T av*	*id*	7, magenta	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4 cl	*S wd*	*ab*	8, green	1	0.92	0.96	1	0.34	0.5	-	-	-	-	-	-
4 cl	*S wl*	*ab*	8, orange	1	0.92	0.96	1	0.34	0.5	-	-	-	-	-	-
4 cl	*S w*	*ab*	8, cyan	1	0.92	0.96	1	0.34	0.5	-	-	-	-	-	-
4 cl	*S av*	*en*	-	0.54	0.94	0.69	0.56	0.28	0.37	-	-	-	-	-	-
4 cl	*S w*	*lr*	9, green	0.83	1	0.9	0.7	0.63	0.67	-	-	-	-	-	-
4 cl	*S w*	*sv*	9, orange	0.78	1	0.88	0.62	0.7	0.67	-	-	-	-	-	-
4 cl	*S w*	*dt*	9, cyan	0.42	1	0.6	0.12	0.45	0.19	-	-	-	-	-	-
4 cl	*S w*	*kn*	9, black	0.5	1	0.67	0.18	0.75	0.29	-	-	-	-	-	-
4 cl	*S w*	*mp*	9, grey	0.78	1	0.88	0.76	0.57	0.65	-	-	-	-	-	-
4 cl	*S w*	*al*	9, magenta	0.6	0.95	0.74	0.17	0.86	0.3	-	-	-	-	-	-
4 cl	*S w*	*as*	9, yellow	0.91	0.98	0.95	0.88	0.46	0.6	-	-	-	-	-	-
4 cl	*S w*	*vl*	9, pink	0.79	0.97	0.88	0.76	0.58	0.66	-	-	-	-	-	-

4 cl	*S w*	*sl*	9, orange	0.76	1	0.86	0.56	0.86	0.68	-	-	-	-	-	-
4 cl	*S w*	*rg*	10, green	1	0.92	0.96	0.97	0.37	0.54	-	-	-	-	-	-
4 cl	*S w*	*rd*	10, orange	1	0.92	0.96	0.94	0.35	0.51	-	-	-	-	-	-
4 cl	*S w*	*rw*	10, cyan	1	0.92	0.96	1	0.36	0.52	-	-	-	-	-	-
100 p	*T tp*	*rd*	11, green	1.0	0.83	0.91	0.82	0.7	0.76	0.05	0.14	0.08	**0.11**	**0.67**	**0.19**

Same legend as in Table 2.

Figure 13 demonstrates a section of the time series; fig. 14, correlation between the number of unforecastable points and errors and the corresponding prediction horizon for using trajectories of possible forecasted values [*rd*]. Fig. 15 shows the same graphs for using a set of possible forecast points. Both graphs correspond to the best ways of calculating final forecasted values and identifying unforecastable points among their respective classes.

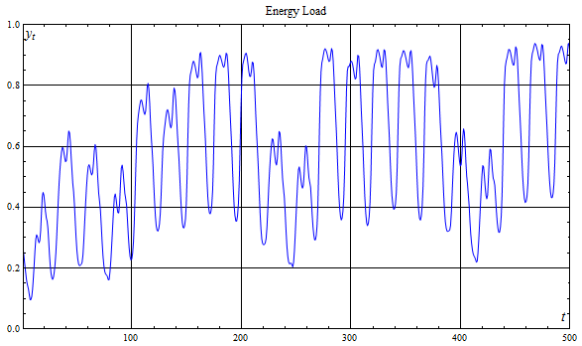
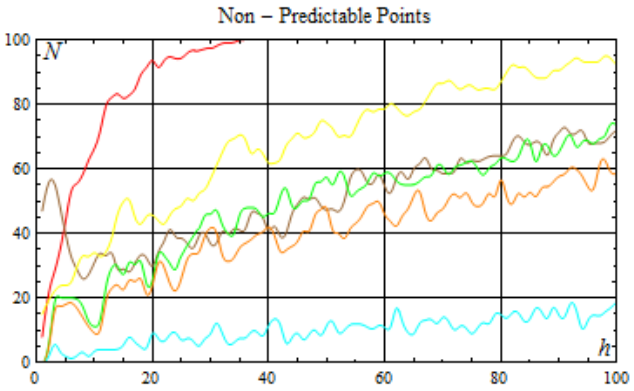


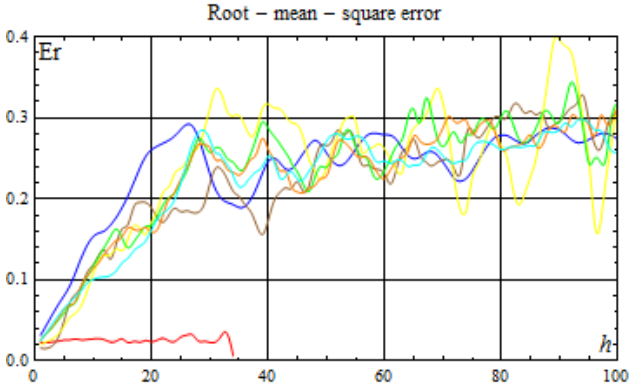
Fig. 13. The energy load series. Typical section

Results obtained in this paper could be compared to the results of [20]. Unfortunately, different integration step sizes and the non-normalized nature of the time series in the aforementioned work do not allow for a direct comparison. However, it is clear that algorithms presented in this work demonstrate good results on a single scroll. After that real and forecasted trajectories start to diverge significantly, while this work has forecasts much further away than a single scroll due to the fact that we do not necessarily have to make a forecast in every point along the way.

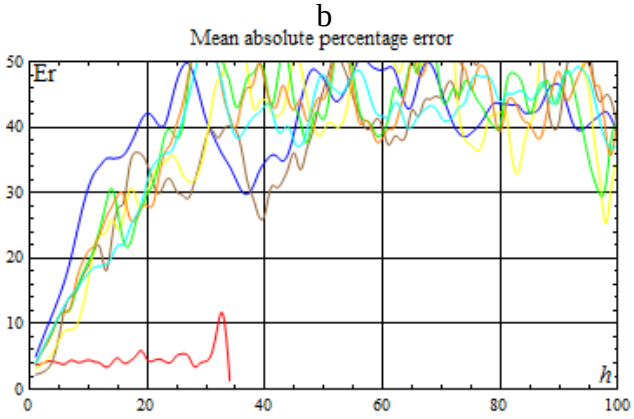
The best method is pointwise using 100% of templates with perturbed trajectories (*100p tp*).



a

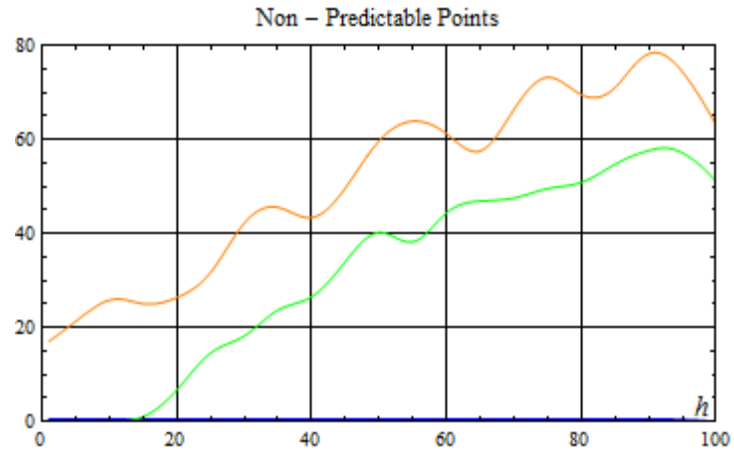


b

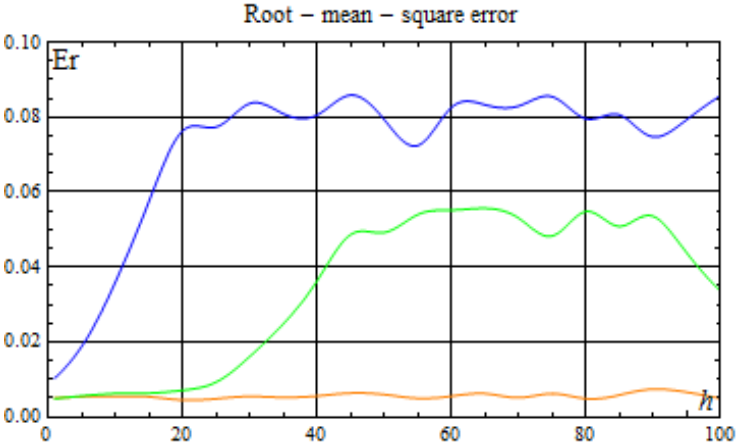


c

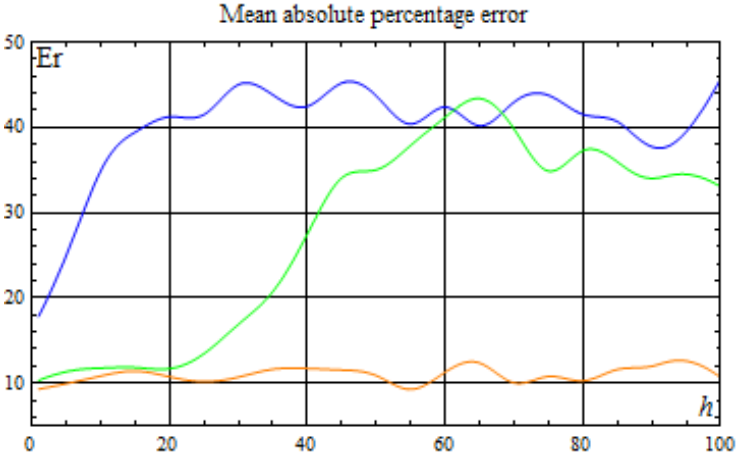
Fig. 15. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon for hourly load values in Germany (4% of patterns are used). The techniques identifies non-predictable points based on growth of the spread [*rg*] (green); of the number of DBSCAN clusters [*sd*] (orange); of the number of Wishart clusters [*sw*] (cyan) over three consecutive points; large entropy values [*en*] (brown); AdaBoost of SVM classifiers[as] (yellow). Red curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points.



a



b



c

Fig. 14. Energy series. The number of non-predictable points (a), RMSE (b), and MAPE (c) vs. a prediction horizon for trajectories with random perturbation (100 % of patterns are used; no clustering technique is applied). The algorithm identifies non-predictable points when possible predicted trajectories diverge. Orange curves correspond to the algorithm that identifies non-predictable points with *a priori* information; blue, to the one that does not identify such points.

4. Conclusions.

Several conclusions may be reached:

1. The paper discusses several novel strategies for multi-step prediction of chaotic time series. Generalized z-vectors, comprising non-successive observations, make it possible to obtain for each point to be predicted, a fairly large set of possible predicted values. If one examines such a set, one may ascertain whether it is possible to produce a unified predicted value for it or not (whether the point is predictable or non-predictable), and determine this unified value, if indeed it is. With non-predictable points, one may state the partial multi-step prediction problem as a two-objective problem. The first functional minimizes the number of non-predictable points, the second, an average error for predictable ones.
2. It appears that for such algorithms the number of non-predictable points grows exponentially with a prediction horizon, but an average error for predictable points remains nearly constant and rather small. The strategies discussed allow a predictive-clustering algorithm to predict positions after a horizon of predictability for a benchmark (Lorenz) and a real-world time series. It was concluded that the exact procedure of calculating the final forecasted value has little effect on the accuracy of forecasting, while an effective method of identifying unforecastable points dramatically expands the horizon of prediction.

3. Large-scale simulation reveals that methods based upon sets of predictable trajectories compare favourably with those based upon sets of predictable points in terms of ultimate prediction quality.
This approach allowed for making some predictions after the prediction horizon of predictability for the corresponding series.

4. The strategies allow one to work with missing data in quite a natural way.

5. Acknowledgements

The authors are deeply indebted to Mr. Joel Cumberland, HSE for the manuscript proof-reading and language editing. Authors are indebted to colleagues from Supercomputer Modelling Unit, HSE for access to the supercomputer and valuable advice.

6. Author Contributions

Vasilii A. Gromov: Supervision, Conceptualization, Methodology, Formal Analysis, Writing

Philip S. Baranov: Formal Analysis, Investigation, Software
Alexandr Yu. Tsybakin: Formal Analysis, Investigation, Software

References

[1]. M. Small, Applied Nonlinear Time Series Analysis: Applications in Physics, Physiology and Finance, Hong Kong Polytechnic University, 2005. https://doi.org/10.1142/5722

[2]. S. Aghabozorgi, A.S. Shirkhorshidi, T.Y. Wah, Time-series clustering – A decade review, Inf. Syst. 23 (2015) 16–38. https://doi.org/10.1016/j.is.2015.04.007

[3]. H. Kantz, T. Schreiber, Nonlinear Time Series Analysis, University Press, Cambridge, 2004. https://doi.org/10.1017/cbo9780511755798

[4]. G.G. Malinetskii, A.P. Potapov, Modern Problems of Non-Linear Dynamics, Editorial URSS, Moscow, 2002.

[5]. B.P. Bezruchko, D.A. Smirnov, Extracting Knowledge From Time Series. An Introduction to Nonlinear Empirical Modeling, Springer, N.-Y., 2010.

[6]. H. Blockeel, L. De Raedt, J. Ramon, Top-down induction of clustering trees, 15th Int. Conf. on Machine Learning (1998) 55–63. https://doi.org/10.1016/S0004-3702(98)00034-4

[7]. V.A. Gromov, E.A. Borisenko, Chaotic time series prediction and clustering methods, Neural Computing and Appl. 2 (2015) 307–315.

[8]. F. Martınez-Alvarez, A. Troncoso, J.C. Riquelme, Energy time series forecasting based on pattern sequence similarity, IEEE Trans. Knowl. Data Eng. 23 (2011) 1230–1243. https://doi.org/10.1109/tkde.2010.227

[9]. S. Ben Taieb, G. Bontempi, A.F. Atiya, A. Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, Expert Syst. Appl. 39 (2012) 7067–7083. https://doi.org/10.1016/j.eswa.2012.01.039

[10]. S. Ben Taieb, A. Sorjamaa, G. Bontempi, Multiple-output modeling for multi-step-ahead time series forecasting, Neurocomputing 73 (2010) 1950–1957. https://doi.org/10.1016/j.neucom.2009.11.030

[11]. V.A. Gromov, Chaotic Time Series Prediction: Run for the Horizon, Proc. of Int. Conf. on Software Test, Machine Learning and Complex Process Analysis (TMPA-2019) Springer (2019) (in press)

[12]. Y. Bao, T. Xiong, Z. Hu, Multi-step-ahead time series prediction using multiple-output support vector regression, Neurocomputing 129 (2014) 482–493. https://doi.org/10.1016/j.neucom.2013.09.010

[13]. M. Sangiorgio, F. Dercole, Robustness of LSTM neural networks for multi-step forecasting of chaotic time series, Chaos, Solitons and Fractals 139 (2020) 110045. https://doi.org/10.1016/j.chaos.2020.110045

[14]. R. Ye, Q. Dai, MultiTL-KELM: A multi-task learning algorithm for multi-step-ahead time series prediction, Appl. Soft Computing J. 79 (2019) 227–253. https://doi.org/10.1016/j.asoc.2019.03.039

[15]. R. Chandra, Y.-S. Ong, C.-K. Goh, Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction, Neurocomputing 243 (2017) 21–34. https://doi.org/10.1016/j.neucom.2017.02.065

[16]. H. Mirzaee, Long-term prediction of chaotic time series with multi-step prediction horizons by a neural network with Levenberg-Marquardt learning algorithm, Chaos, Solitons and Fractals 41 (2009) 1975–1979. https://doi.org/10.1016/j.chaos.2008.08.016

[17]. V.A. Gromov, A.N. Shulga, Chaotic time series prediction with employment of ant colony optimization, Expert Syst. with Appl. 39 (2012) 8474–8478. https://doi.org/10.1016/j.eswa.2012.01.171

[18]. V.A. Gromov, A.S. Konev, Precocious identification of popular topics on Twitter with the employment of predictive clustering, Neural Computing and Appl. 28 (2017) 3317–3322. 10.1007/s00521-016-2256-1

[19]. V.A. Gromov, I.M. Voronin, V.R. Gatylo, E.T. Prokopalo, Active Cluster Replacement Algorithm as a Tool to Assess Bifurcation Early-warning Signs for von Karman equations, Artificial Intelligence Res. 6 (2017) 51–56. https://doi.org/10.5430/air.v6n2p51

[20]. S. Kurogi, R. Shigematsu, K. Ono, Properties of Direct Multi-Step Ahead Prediction of Chaotic Time Series and Out-of-Bag Estimate for Model Selection, C.K. Loo, K.S. Yap, K.W. Wong, A. Teoh, K. Huang (eds) Neural Information Processing. ICONIP 2014. Lecture Notes in Computer Science 8835 (2014), Springer, Cham. https://doi.org/10.1007/978-3-319-12640-1_51

[21]. W. Waheeb, R. Ghazali, Multi-step Time Series Forecasting Using Ridge Polynomial Neural Network with Error-Output Feedbacks, Berry M., Hj. Mohamed A., Yap B. (eds) Soft Computing in Data Science. SCDS 2016. Communications in Computer and Information Science, 652 (2016) Springer, Singapore. https://doi.org/10.1007/978-981-10-2777-2_5

[22]. P. Ong, Z. Zainuddin, Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction, Appl. Soft Computing J. 80 (2019) 374–386. https://doi.org/10.1016/j.asoc.2019.04.016

[23]. R. Wang, C. Peng, J. Gao, Z. Gao, H. Jiang, A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series, Comp. and App. Math. 39 (2020) 30. https://doi.org/10.1007/s40314-019-1006-2

[24]. D.R. Cox, Prediction by exponentially weighted moving averages and related methods, J. R. Stat. Soc. B 23 (1961) 414–422. https://doi.org/10.1111/j.2517-6161.1961.tb00424.x

[25]. G. Bontempi, Long term time series prediction with multi-input multi-output local learning, Second European Symposium on Time Series Prediction (2008) 145–154.
[26]. K. Wang, K. Li, L. Zhou, Y. Hua, Z. Cheng, J. Liu, C. Chen, Multiple convolutional neural networks for multivariate time series prediction, Neurocomputing 360 (2019) 107–119. https://doi.org/10.1016/j.neucom.2019.05.023
[27]. D. Wishart, A numerical classification methods for deriving natural classes, Nature 221 (1969) 97–98. https://doi.org/10.1038/221097a0
[28]. H.H. Bock, Automatic Classification, Vandenhoeck and Rupert, Göttingen, 1974. https://doi.org/10.1002/bimj.4710200111
[29]. A.V. Lapko, S.V. Chentsov, Nonparametric Information Processing Systems, Nauka, Novosibirsk, 2000.
[30]. C.C. Aggarval, C.K. Reddy (Eds.), Data Clustering. Algorithms and Applications, CRC Press. Taylor & Francis Group, Boca Raton, Florida, 2014. https://doi.org/10.1007/s00362-015-0661-7
[31]. N.V. Chawla, K. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, J. of Artificial Intelligence Res. 16 (2002) 321–357. https://doi.org/10.1613/jair.953
[32]. E.A. Jackson, The Lorenz System: I. The Global Structure of its Stable Manifolds, Phys. Scr. 32 (1985) 469–475. https://doi.org/10.1088/0031-8949/32/5/001
[33]. O.A. Rosso, H.A. Larrondo, M.T. Martin, A. Plastino, M.A. Fuentes, Distinguishing Noise from Chaos, Phys. Rev. Letters, 99 (2007) 154102.
[34]. J.P. Eckmann, S.O. Kamphorst, D. Ruelle, S. Ciliberto, Liapunov exponents from time series, Phys. Rev. A 34 (1986) 4971–4979. https://doi.org/10.1103/physreva.34.4971

Appendix A

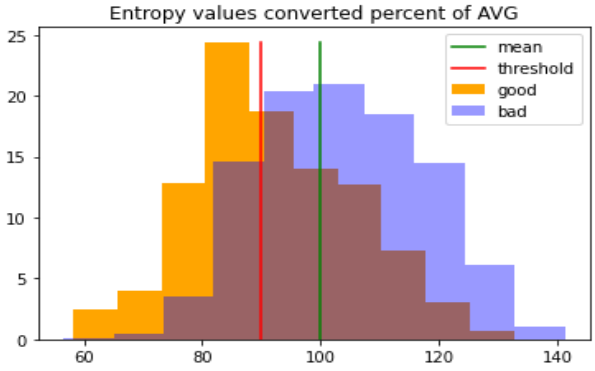


Fig. A1. A typical empirical distribution for entropy for the ground-truth non-predictable points (orange) and its complement to a testing set (blue) for the Lorenz series; brown colour denotes an intersection. Green line stands for a mean value for non-predictable points; red, for a threshold between predictable and non-predictable points.

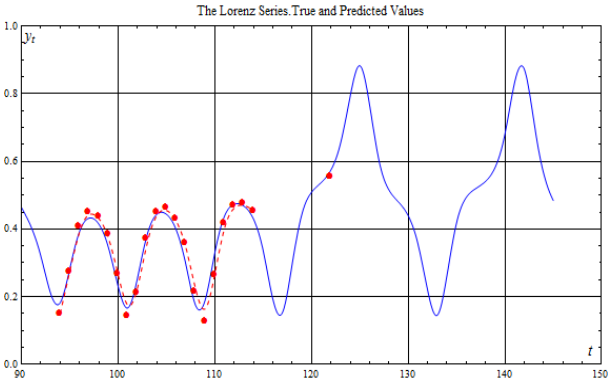


Fig. A2. The Lorenz series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm identifies non-predictable points using logistic regression [*lr*].

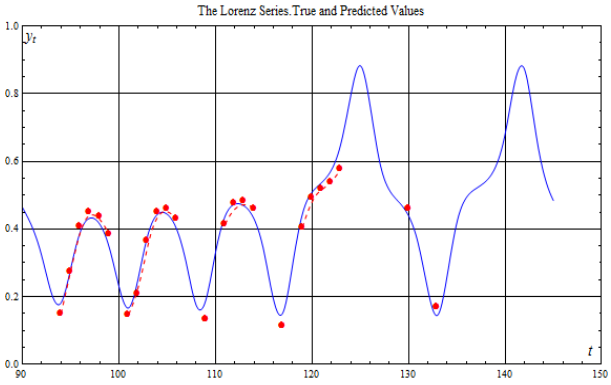


Fig. A3. The Lorenz series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm identifies non-predictable points using multi-layer perceptron [*mp*].

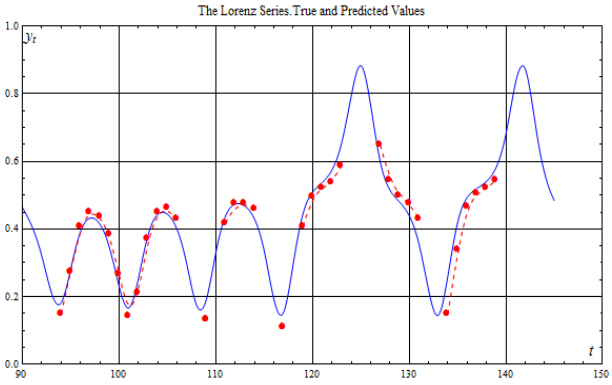


Fig. A4. The Lorenz series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm identifies non-predictable points using AdaBoost that combines support vector machines [*as*].

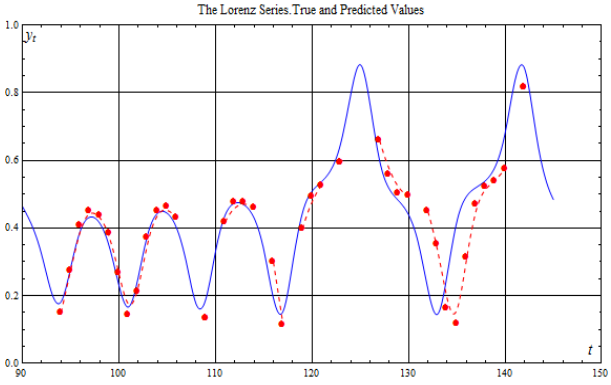


Fig. A5. The Lorenz series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm identifies non-predictable points using stacking of logistic regressions [*sl*].

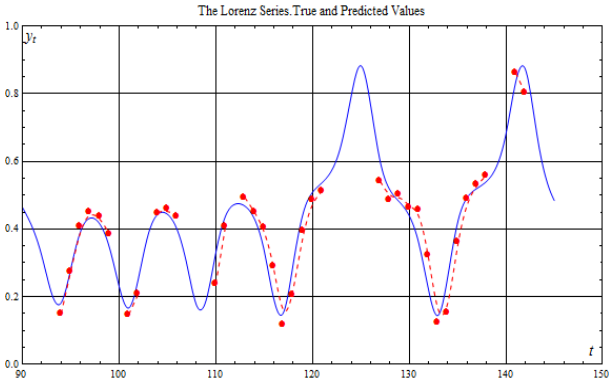


Fig. A6. The Lorenz series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm identifies non-predictable points using growth of spread [*rg*].

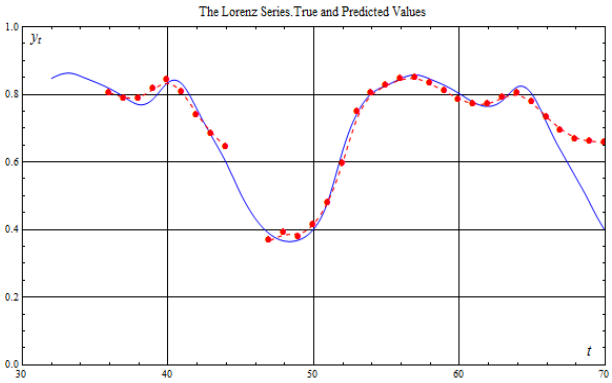


Fig. A7. German energy time series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm identifies non-predictable points using AdaBoost of SVM classifier [*as*].

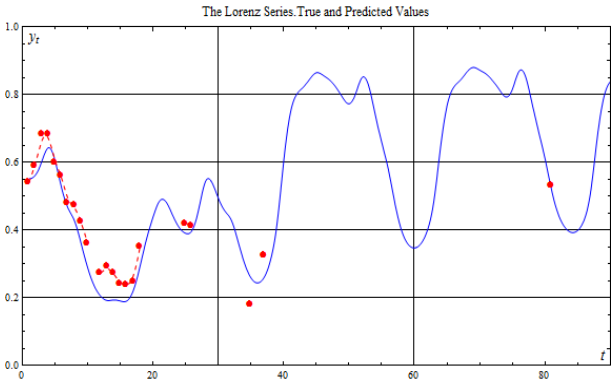


Fig. A8. German energy time series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm employs possible predicted trajectories [*T*] identifies non-predictable points using spread growth [*rg*].

Appendix B

Table B1. Abbreviations for methods of *A training set*

p	**Pointwise**	1
cl	**Cluster**	2

Table B2. Abbreviations for methods of *A unified predicted value* (algorithm types)

T	**Trajectory**. One uses a set of possible predicted trajectories	1
S	**Set**. One uses a set of possible predicted values	2

Table B3. Abbreviations for methods of *A unified predicted value* (calculation methods)

av	**Average.** The value is an average	1.1
wl	**Weighted sum, length.** The value is a weighted average with weights determined by a prediction length	1.2
wd	**Weighted sum, distance.** Weights are determined by a distance to a cluster centre	1.2
w	**Weighted sum, combined.** Weights are determined by a product of the two previous weights	1.2
cm	**Cluster max.** The value is a centre of the largest cluster	1.3
cw	**Cluster max, weighted.** The value is a centre of the largest cluster, and clustering is performed for possible predicted values with relatively large weights	1.4
fs	**Fuzzy set.** Denotes fuzzy clustering	1.5
rw	**Roulette wheel.** The value is the centre of the cluster chosen probabilistically	1.6
mf	**Most frequent.** The value is the most frequent possible predicted value	1.7
mp	**Most frequent, perturbed.** The most frequent possible predicted value with perturbation	1.8
tp	**Trajectory, perturbed.** The value is an average over the last points of the perturbed trajectories	2.1
tm	**Trajectory, multiple clustering.** The value is an average over the last points of the branching trajectories	2.2
tr	**Trajectory, reduced information.** The value is an average over the last points of the trajectories that rely on reduced initial information	2.3

Table B4. Abbreviations for methods of *Algorithms to identify non-predictable points*

fp	**Forced prediction**. Non-predictable points are not identified altogether, predicted values are forcibly calculated at all intermediate points	-
ab	**Absent.** Implies that a set of non-predictable points consists only of the points that the algorithm failed to find motifs close enough, sets of possible predicted values are not analysed (the first extreme case)	-
id	**Ideal.** Sets of possible predicted values are constructed using *a priori* information (the second extreme case)	-
en	**Entropy.** Implies that predictable and non-predictable points are separated using entropies of their possible predicted values distributions	-
lr	**Logistic regression.** One applies the classifier to features of possible predicted values sets	1.1
sv	**Support vector machine.** One applies a support vector machine	1.2
dt	**Decision tree.** One applies a decision tree C4.5	1.3
kn	**K-nearest neighbours.** One applies a k-nearest neighbours classifier	1.4
mp	**Multi-layer perceptron.** One applies a multi-layer perceptron	1.5
al	**AdaBoost logistic regression.** Classifiers (logistic regressions) are assembled using AdaBoost	2.1
as	**AdaBoost SVM.** Classifiers (SVM) are assembled using AdaBoost	2.1
sl	**Stacking logistic regression.** Classifiers (logistic regression) are assembled by stacking	2.2
vl	**Voting logistic regression.** Classifiers (logistic regression) are assembled by voting	2.3
ls	**Large spread.** Implies that the point is non-predictable if the spread of its possible predicted values exceeds an average spread	3.1
rg	**Rapid growth.** The spread grows monotonically at several consecutive points	3.2
rd	**Rapid growth. DBSCAN.** A monotonic growth of the number of possible predicted value clusters	3.2

	obtained with DBSCAN	
rw	**Rapid growth. The Wishart clustering.** A monotonic growth of the number of possible predicted value clusters obtained with the Wishart clustering	3.2
ca	**Compare with average.** One compares the principle predicted trajectory with a weighted average of other predicted trajectories	3.3
ad	**Average, difference.** One calculates an average of modulus of a difference	3.4
wa	**Weighted average**. One calculates a weighted average of modulus of a difference	3.5
wd	**Weighted compare.** One calculates a weighted average of modulus of a difference	3.6