Hindawi
Complexity
Volume 2023, Article ID 6689371, 21 pages
https://doi.org/10.1155/2023/6689371

Research Article

Prediction after a Horizon of Predictability: Nonpredictable Points and Partial Multistep Prediction for Chaotic Time Series

Vasilii A. Gromov and Philip S. Baranov

HSE University, Pokrovsky Boulevard 11, Moscow, Russia

Correspondence should be addressed to Vasilii A. Gromov; stroller@rambler.ru

Received 6 February 2023; Revised 11 May 2023; Accepted 21 August 2023; Published 21 September 2023

Academic Editor: Hiroki Sayama

This paper introduces several novel strategies for multi-step-ahead prediction of chaotic time series. Introducing a concept of "generalized z-vectors" (vectors of nonsuccessive time series observations) makes it possible to generate sets of possible prediction values for each point we are trying to predict. Through examining these sets, unified predictions are calculated, which are in turn used as a basis for predicting subsequent points. The key difference between the strategy presented in this paper and its conventional counterparts is the concept of "nonpredictable" points (points which the algorithm categorized as "incalculable" and excluded from the calculations altogether). The results obtained for the benchmark and real-world time series indicate that while typically the number of nonpredictable points tends to grow exponentially with the number of steps ahead to be predicted, the average error for predicted points remains small and nearly constant. Thus, we redefine the problem of multi-step-ahead prediction as a two-objective optimization problem: on one hand, we aim to minimize the number of nonpredictable points and the average error among the predictable ones. The resulting strategy demonstrates accurate results for both benchmark and real-world time series, with the number of predicted steps exceeding that of any other published algorithm.

1. Introduction

There are countless examples of chaotic systems in the world, which is evidenced by the constantly increasing number of chaotic time series forecasting algorithms. However, while those dealing with one-step-ahead prediction demonstrate remarkable efficiency [1], their multi-step-ahead prediction counterparts are still in their infancy. This may be attributed to the exponential growth of an average prediction error with increasing prediction horizon (the number of steps ahead to be predicted).

This exponential growth reflects Lyapunov instability, inherent to any chaotic system. According to the definition of *Lyapunov instability*, any initial difference between any two neighboring trajectories, however small, grows exponentially over time; its exponent equals to the highest Lyapunov exponent λ [2, 3]. Lyapunov instability leads to another concept, that of the *horizon of predictability*, sometimes referred to as "Lyapunov time" [4]. For a given observational error $\varepsilon(0)$ and the maximum prediction error $\varepsilon_{\max}$, the aforementioned exponential growth satisfies the constraint $\varepsilon(t) = \varepsilon(0)e^{\lambda t}$, where $\varepsilon(t)$ represents error at the moment t and λ, the highest Lyapunov exponent (which is positive for chaotic time series and negative for the nonchaotic ones). $\varepsilon(t) \leq \varepsilon_{\max}$ gives an estimated horizon of predictability $T \sim 1/\lambda \ln\varepsilon_{\max}/\varepsilon(0)$ [2, 3]. In the context of the multi-step-ahead prediction, it follows that the smallest difference between true and predicted values at any intermediate position between the last observable point and the point to be predicted triggers an exponential error growth in all subsequent points, regardless of the employed prediction method. It should be noted that the *horizon of predictability* should not be confused with the *prediction horizon*, the former being a theoretical upper boundary for the number of steps to be predicted (given $\varepsilon(0)$ and $\varepsilon_{\max}$), and the latter being simply the number steps to be predicted. Most of the time, the *prediction horizon* h is significantly less than the *horizon of predictability* T: $h \ll T$. However, predictive clustering approach provides the necessary tools for dealing with exponential growth of the average prediction error [5, 6].

First, predictive clustering utilizes *motifs*, repeated sequences of data in a series. If a section of a time series resembles an initial part of a *motif* "closely enough," it is likely that the subsequent points of the series will closely match the subsequent points of the *motif*, thus enabling the use of *motifs* for forecasting the time series at hand. *Motifs* can be obtained by clustering vectors of observations and calculating their centers. It should be noted that whereas most available prediction methods attempt to construct a single unified prediction model, the *motifs* provided by predictive clustering form a set of local prediction models.

Second, Gromov and Borisenko [6] proposed using generalized z-vector *templates*, vectors of nonsuccessive observations, which are spaced out according to some predefined *pattern*. A *pattern* is a vector of distances between positions of series observations. The resulting vector generalizes a conventional *z-vector* (successive observations), which corresponds to a pattern $(1, 1, \ldots, 1)$ $L - 1$ times. Figure 1 demonstrates a z-vector constructed according to a pattern $k_1, k_2, \ldots, k_{L-1}, k_i \in \mathbb{N}$, whereas Figure 2 illustrates the way z-vectors are clustered into *motifs* and also how *motifs* are used for obtaining predictions. The process of generating z-vectors can be visualized as placing a "comb" with L teeth spaced out according to some pattern, with all the other teeth "broken off." While moving the "comb" along the series, we can obtain a vector of observations under the teeth of the "comb" $(y_t, y_{t+k_1}, \ldots, y_{t+k_1+\ldots+k_{L-1}})$ for each position t. Each pattern $k_1, k_2, \ldots, k_{L-1}, k_i \in \mathbb{N}$ has its own set of such vectors (the sample corresponding to a given pattern). Each such a sample is then clustered separately. This approach obtains a set of possible values for each point to be predicted. Each set is comprised of predicted values obtained using motifs corresponding to different templates. Since the number of patterns can be arbitrarily large, so can sizes of sets of possible prediction values for points we are trying to predict. Despite the fact that points in those sets are not always statistically independent, a great deal of algorithms determining a unified predicted value based on these sets can be designed.

Third, the concept of nonpredictable points can be extended. However, previously a point was considered to be nonpredictable if it did not have corresponding motif(s), and now it can also be considered nonpredictable if it is impossible to calculate a unified predicted value for this particular point based on its set of possible prediction values. An example of the latter would be a point which set of possible prediction values consists of two equally-sized clusters with different centers. Present work introduces a novel concept of nonpredictable points that the authors have not yet encountered in any literature on the topic. It should be noted that while in our previous paper, we employed only the former type of nonpredictable points, but we use both in this one.

Introduction of a concept of *non-predictable points* renders the problem two-objective: on one hand, we have to minimize the total number of *non-predictable points*, and on the other, the average error among the predictable ones. Discarding some of points as nonpredictable has proven to have a positive effect on the results, and it is better for the algorithm to skip some of the points rather than force a prediction at each point. This approach is actually quite natural in some areas: for example, stock market traders do not have to make trades at every single moment in time, as they could simply choose the moments that the algorithm had recognized as predictable.

Furthermore, since the patterns $k_1, k_2, \ldots, k_{L-1}, k_i \in \mathbb{N}$ with distance $d = k_{L-1} - k_{L-2} > 1$ exist, the fact that the point at position t is nonpredictable does not imply that the points at subsequent positions $t + 1, t + 2, \ldots$ will be nonpredictable as well. This is fundamentally important for the multi-step-ahead prediction: if a point $t + h$ needs to be predicted, iterative strategy would be used (or rather its modifications based on nonsuccessive observations). Namely, the intermediate values between points t and $t + h$ are predicted step-by-step, with nonpredictable points being identified along the way and ignored.

In the context of dynamic systems [2, 3], each cluster of vectors (which by definition represent similar sections of a trajectory) corresponds to a particular area of a strange attractor. Areas with a high value of an invariant measure are associated with larger clusters (i. e., more frequently observed motifs), and vice versa. The number of clusters increases with the size of their respective training set, whereas the number of *nonpredictable points* and the average error among the predictable ones decrease. A large-scale simulation for the Lorenz series [7] supports this conclusion.

In conclusion, however, the algorithm does not calculate a prediction for each point (thus the word "partial" in the title), it does make predictions up to the horizon of predictability (sometimes even surpassing it). The paper presents several methods of identifying nonpredictable points and corresponding strategies for the multi-step-ahead prediction, as well as examples of partial predictions beyond the horizon of predictability, for the benchmark and real-world time series.

The rest of the paper is organized as follows:

(i) Section 2 reviews recent advances in the field

(ii) Section 3 formally states the problem

(iii) Section 4 outlines the employed clustering technique and ways of identifying nonpredictable points and calculating predictions

(iv) Section 5 provides the results for predicting the Lorenz series and hourly electric grid load values in Germany

(v) Section 6 contains conclusions

2. Related Works

Most of the recently published papers discussing chaotic time series prediction [1, 6, 8–11] concern themselves only with the one-step-ahead prediction problem, whereas the number of papers tackling the problem of multi-step-ahead prediction (MSAP) is considerably lower.

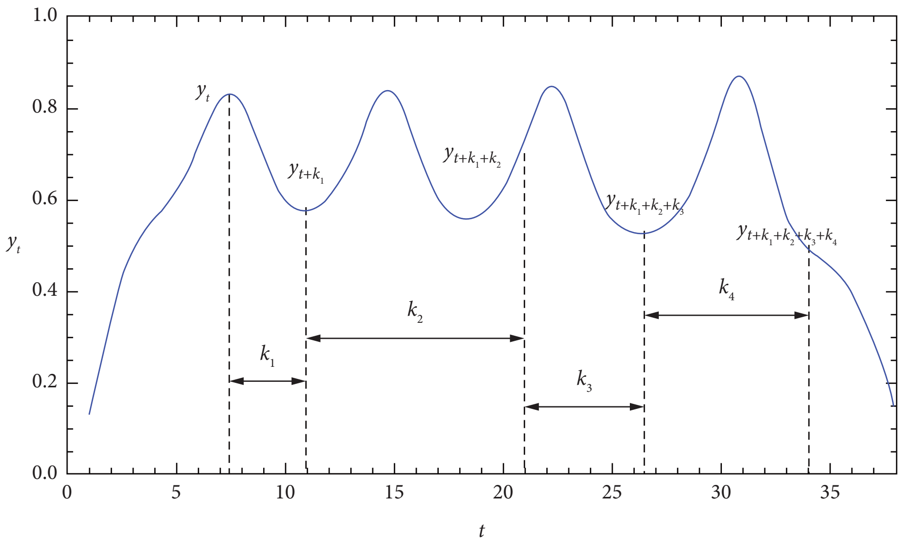

Figure 1: A sample z-vector $(y_t, y_{t+k_1}, y_{t+k_1+\ldots+k_{L-1}})$ concatenated for a pattern $k_1, \ldots, k_{L-1}, k_i \in \mathbb{N}$.

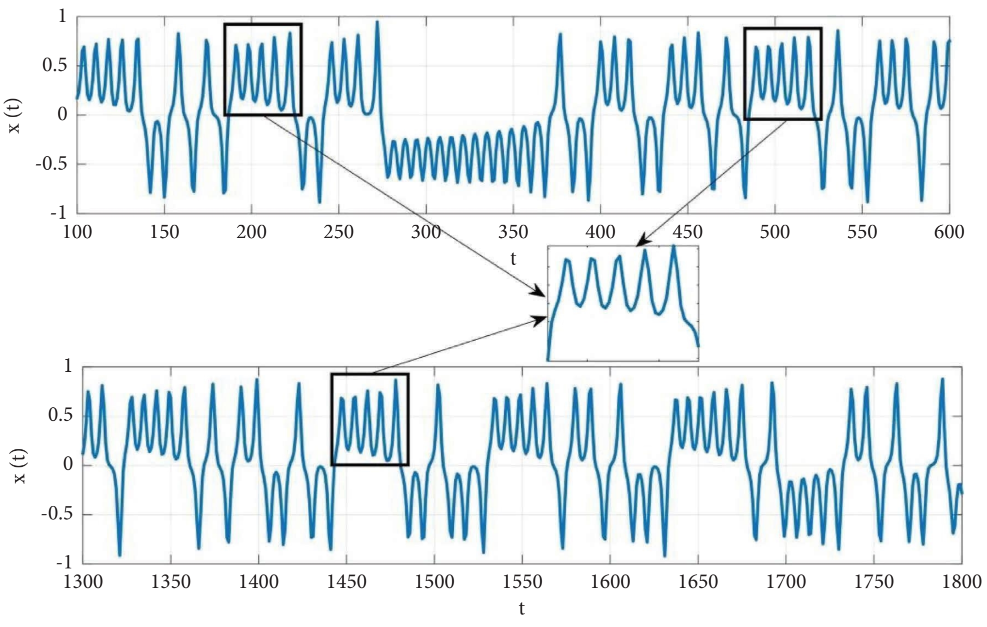

Figure 2: Representation of the way the algorithm finds similar sections in a series (top), clusters them (middle), and uses the center of the obtained cluster to make predictions for the test set (bottom).

An MSAP algorithm for chaotic time series consists of two parts, which are equally responsible for the accuracy of the final prediction. The first part is a technique used to make one- or few-steps-ahead predictions, whereas the second one "assembles" the results of the previous part into a final multi-step-ahead prediction.

Algorithms used to make a one-step-ahead prediction employ nearly all fields of data mining and machine learning, such as

(i) Support vector regression and its modifications [9]

(ii) Multilayer perceptron [12]

(iii) Clustering algorithms in predictive clustering [13–16]

(iv) LSTM neural networks [10]

(v) Voronoi diagrams [17]

(vi) Ridge polynomial neural networks [18]

(vii) Wavelet neural networks [19, 20]

(viii) Dilated convolution networks [21]

Another factor of fundamental importance is a strategy that "assembles" one-step-ahead predictions into a multi-step-ahead one. It involves two basic strategies of MSAP, the iterated (recursive) and direct strategies [22, 23]. The iterated one implies that multi-step-ahead prediction is a process carried out step-by-step: new predicted values are calculated based on the predictions made in the previous steps. The direct strategy aims for getting the results immediately without calculating predicted values for intermediate positions. This strategy applies prediction techniques for various prediction horizons, thereby providing multiple predictions for any position to be predicted. Ben Taieb et al. [22] review different methods based on these two basic strategies. Sangiorgio and Dercole [10] apply both strategies with multilayer perceptrons and LSTM nets employed as tools to make one-step-ahead predictions.

Unfortunately, MSAP methods designed in the aforementioned strategies suffer from the same exponential error growth, thus giving rise to a number of hybrid strategies aiming to resolve it. In their review, Ben Taieb et al. [8] compare the two basic and three novel strategies (DirRec, MIMO, and DIRMO). DirRec (Direct + Recursive) combines the two basic strategies and uses the direct approach to predict values with the number of inputs being enlarged iteratively to include values of the most recently predicted positions. When it comes to MIMO (multiple input multiple output) strategy [24, 25], an array of values is produced for the positions between the observed values and a prediction horizon (inclusively). This reveals any correlations the time series may have and allows them to be stored within predicted values.

DIRMO (DirRec + MIMO) strategy divides the series (up to some prediction horizon) into chunks and applies MIMO strategy to each chunk separately. The authors test these five strategies on a reasonably large sample of different time series (NN5 competition), which reflects various irregularities inherent to time series. Bao et al. [9] compare efficiency of the iterated, direct, and MIMO strategies by performing a one-step-ahead prediction using a modified support vector regression. According to the authors, the MIMO strategy compares favourably with the other two (all other factors being equal).

Chaotic time series prediction methods that rely on reservoir computing would mark a different approach [26, 27]. Canaday et al. indicate that the method demonstrates excellent results for small prediction horizons; however, their performance worsens greatly when applied to larger horizons [28].

Multiple-task learning can be viewed as a strategy of its own for multi-step-ahead prediction. Chandra et al. [12] propose an algorithm to determine a neural network structure to solve MSA prediction problems; their approach can be considered a combination of the direct and iterated strategies. Wang et al. [29] utilise a neural model in order to combine periodic approximations for longer periods and machine learning models for the shorter ones. This could be viewed as a separate strategy when dealing with data with a marked periodicity. Ye and Dai [11] employ multitask learning for multistep prediction. Kurogi et al. [17] make use of an out-of-bag model for selecting models for multistep prediction. The authors aim at predicting chaotic series as far as possible (the largest possible prediction horizon), whereas retaining reasonable accuracy. The authors present their results for the Lorenz series.

The importance of being able to make accurate predictions up to an increasing number of steps was named "Run for the Horizon" by the authors of the current paper in one of their previous publications [7]. The series of works by Sangiorgio et al. [10, 30] (that culminated in a monograph [25] in 2021) approach multi-step-ahead forecasting by using neural networks, classic feed-forward, and recurrent architectures (LSTM) nets. The latter are traditionally trained with a supervisor, thus forcing the algorithm the speed up the convergence of the optimization. The authors managed to make adequate predictions up to six Lyapunov times on the benchmark series (logistic and Hénon maps as well as two generalized Hénon maps). Even though the authors did manage to considerably delay the exponential error growth, they failed to avoid it completely; the results indicate that after six Lyapunov times the error starts to increase exponentially. It should also be noted that the predictions achieved in [31], which relied on reservoir computing with the data calculated by the integration of the Lorenz-96 model, are similar in terms of prediction intervals.

To summarize, none of the aforementioned strategies are immune to the exponential growth of an average prediction error with an increasing prediction horizon. The present paper discusses a few novel strategies with the main difference from their classical counterparts being the concept of nonpredictable points and an ability not to take into account clearly erroneous predictions at intermediate positions [6], thus weakening the exponential nature of the growth rate.

3. Problem Statement

This paper deals with an h-steps-ahead prediction for a chaotic time series $Y = \{y_0, y_1, \ldots, y_t, \ldots\}, h > 0 \in \mathbb{N}$. We assume that all transient processes are completed, and that the series itself reflects the movement of a trajectory in the neighborhood of a strange attractor. The third assumption is that the series meets the conditions of the Takens' theorem (which makes the analysis of the attractor structure based on the series observations possible) [2, 3].

We divide the series into two parts: $Y_1 = Y_1(t) = \{y_0, y_1, \ldots, y_t\}$, the observable part used as a training set, and the test set $Y_2 = Y_2(t) = \{y_{t+1}, y_{t+2}, \ldots, y_{t+h}, \ldots\}$, $Y = Y_1 \cup Y_2, Y_1 \cap Y_2 = \varnothing$. When the algorithm makes a prediction at a position $t + h$ of the test set, is has access to observations $[y_{t-s+1}, y_{t-s+2} \ldots, y_t]$; however, it does not have any information on observations $[y_{t+1}, y_{t+2}, \ldots y_{t+h-1}]$. Thus, $\widehat{y}_{t+h} = \widehat{y}_{t+h}(\{y_t, y_{t-1} \ldots, y_{t-s+1}\})$, where s is a parameter of the algorithm.

Predictive clustering algorithms and a large number of used patterns make it possible to construct a set of possible predicted values $\widehat{S}_{t+h} = \{\widehat{y}_{t+h}^{(1)}, \ldots, \widehat{y}_{t+h}^{(N_{t+h})}\}$ for each point to be predicted, where N_{t+h} is the number of possible predicted values calculated by the algorithm and $y_{t+h}^{(i)}, i = 1..N_{t+h}$ is the i-th predicted value. A set $\widehat{S}_{t+h}^{(p)} = \{\widehat{S}_{t+1}, \ldots, \widehat{S}_{t+h}\}$ is comprised of all sets of possible predicted values for the points $[y_{t+h-p}, y_{t+h-p+1}, \ldots, y_{t+h}]$, where p indicates the type of algorithm employed (usually p is equal to 1 or h). For $p = 1$, $\widehat{S}_{t+h}^{p}$ consists of sets of possible predicted values for the point

y_{t+h} exclusively; for $p = h$, it consists of sets of possible predicted values for the points $[y_t, y_{t+1}, \ldots, y_{t+h}]$. We denote the algorithm that constructs sets of possible predicted values $\widehat{S}_{t+h}$ as f_h: $\widehat{S}_{t+h} = f_h(\{y_t, \ldots, y_{t-s+1}\})$.

The concept of nonpredictable points implies that two operators are applied to the set $\widehat{S}^{(p)}_{t+h}$. The first checks if a position is predictable:

$$\zeta\left(\widehat{S}^{(p)}_{t+h}\right) = \begin{cases} 1, & \text{if position is predictable,} \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

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

The second operator determines a unified predicted value for a set of possible predicted values (provided the position is predictable):

$$\widehat{y}_{t+h} = g\left(\widehat{S}^{(p)}_{t+h}\right). \tag{2}$$

Using these two operators, we can define the multi-step-ahead prediction process as a twofold optimization problem:

$$\begin{aligned} I_1 &= \sum_{t+h \in Y_2} \left(1 - \zeta\left(\widehat{S}^{(p)}_{t+h}\right)\right), \\ I_2 &= \frac{1}{|Y_2|} \sum_{t+h \in Y_2} \zeta\left(\widehat{S}^{(p)}_{t+h}\right) \left\| g\left(\widehat{S}^{(p)}_{t+h}\right) - y_{t+h} \right\|. \end{aligned} \tag{3}$$

Both functionals sum over all observations of the test set, with first one minimizing the total number of nonpredictable points and the second one minimizing the average error among the predictable ones. The present paper attempts to solve this two-objective problem.

3.1. Algorithm.

3.1. Algorithm. The subsection is organized as follows:

(i) The process of generating samples from the time series according to predefined patterns
(ii) Employed clustering techniques
(iii) Generating sets of possible prediction values
(iv) Quality measures of identification of nonpredictable points
(v) Ways of identifying nonpredictable points

3.2. Training Set. The series is assumed to be normalized. Patterns (arrays of distances between nonsuccessive observations) are used to generate samples. Each pattern is an $L-1$-dimensional vector of integers $(k_1, k_2, \ldots, k_{L-1})$, $1 \le k_j \le K_{\max}$, where $K_{\max}$ is the maximum distance between any two consecutive observations in the pattern.

For example, let us consider a three-point pattern (2,3,4). The first vector of the corresponding training set would be (y_0, y_2, y_5, y_9) (the first z-vector); the next vector would then become (y_1, y_3, y_6, y_{10}) and so forth. The last vector of the training set would be $(y_{t-9}, y_{t-7}, y_{t-4}, y_t)$ with y_t being the last observable value of the training set.

In the context of predictive clustering, the conventionally used z-vectors comprising successive observations have proven to be less efficient than their counterparts comprising nonsuccessive observations [6]. We attribute it to the fact that vectors of nonsuccessive observations have more chances to store information about salient observations (minima, maxima, tipping points, etc.) and correlations between them.

A set of all combinatorically possible patterns $\aleph(L, K_{\max})$ is used for generating training sets. Each set is generated separately, according to corresponding pattern $\alpha \in \aleph(L, K_{\max})$. Vectors from these training sets can be used either "as-is" (i.e,. each vector is treated as a separate motif), or they can be clustered first, with the center of each of the formed clusters becoming a motif of its own.

We denote a set of motifs corresponding to the pattern $\alpha = (k^\alpha_1, k^\alpha_2, \ldots, k^\alpha_{L-1}), \alpha \in \aleph(L, K_{\max})$ as $\Psi_\alpha = \{C_\alpha\}, C_\alpha = (\eta^\alpha_1, \eta^\alpha_2, \ldots, \eta^\alpha_L)$. The obtained motifs can only be used with the pattern α. The set of all motifs is denoted as $\Psi = \{(\alpha, \Psi_\alpha)\}, \alpha \in \aleph(L, K_{\max})$.

3.3. Clustering Technique. As a trajectory moves repeatedly through the same area of a strange attractor, similar recurring sequences of values associated with that particular area can be observed in the trajectory's time series. Centers of clusters of sequences associated with different areas of the attractor serve as simplest prediction models for the respective areas of the attractor [14]. We used the clustering method outlined below to cluster sequences of observations. We employ a modified version of the Wishart clustering technique [32, 33] developed by Lapko and Chentsov [34]. It uses graph theory concepts and a nonparametric probability density function estimator of r-nearest neighbors. Some of the difficulties associated with the application of this algorithm to forecasting are discussed in [6].

A significance value of a point x is defined as $p(x) = r/V_r(x)n$, where $V_r(x)$ and $d_r(x)$ are the volume and the radius of a minimum-size hypersphere containing at least r observations, centered at point x (n is the number of sample vectors). The method relies on a proximity graph $G(Z_n, U_n)$, vertices of which 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 subgraph of $G(Z_n, U_n)$ with a vertex set $Z_q = \{x_j, j = 1..q\}$ and an edge set comprised of all edges of U_n in a such a way that its vertices belong to Z_q. Let $w(x_q)$ be a cluster number label of x_q. A cluster $c_l, l > 0$ is said to be height-significant with respect to the height value $\mu > 0$ if $\max_{x_i, x_j \in c_l}\{|p(x_i) - p(x_j)|\} \ge \mu$ Algorithm 1.

Thus, the algorithm consists of the following steps:

After performing large-scale simulations for different parameter values, we determined the best values of r and μ to be 11 and 0.2, respectively.

```
determine $d_r(x_q)$ = distance to the sample's $r$-nearest neighbor;
sort $d_r(x_q)$ in ascending order;
$q = 1$;
for each subgraph $G(Z_q, U_q)$:
  while $q \le n$:
  $x_q$ = newly added vertex of the subgraph;
  if $x_q$ is not connected to any clusters:
    start new cluster;
  else:
    if $x_q$ connected to the vertices of clusters $c_1, c_2, \ldots, c_l, l \ge 1$:
      if all clusters are completed:
        $w(x_q) = 0$;
      else:
        $k(\mu)$ = number of significant clusters;
        if $k(\mu) > 1$ or $c_1 = 0$:
          $w(x_q) = 0$;
          label significant clusters as completed;
          delete labels of insignificant clusters;
        else:
          merge clusters $c_2, \ldots, c_n$ into $c_1$;
          $w(x_q) = c_1$;
          set $w(x_i) = c_1$ for samples in $c_2, \ldots, c_n$;
    $q = q + 1$;
```

Algorithm 1: The Wishart clustering algorithm.

3.4. Sets of Possible Predicted Values. We consider a set of motifs Ξ_α for each pattern $\alpha \in \aleph(L, K_{\max}), \alpha = (k_1^\alpha, \ldots, k_{L-1}^\alpha)$ and construct a set of possible prediction values $\widehat{S}_{t+h}^{p,\alpha}$ associated with the pattern α. Namely, we compose a vector of time series observations corresponding to the pattern α: $C = (y_{t+h-k_{L-1}^\alpha}, y_{t+h-k_{L-1}^\alpha-k_{L-2}^\alpha}, \ldots, y_{t+h-k_{L-1}^\alpha-k_{L-2}^\alpha-\ldots-k_1^\alpha})$ for a position $t + h$ we are trying to predict (all elements of C are assumed to be either observed or predicted in the previous step(s)). The next step is to calculate the Euclidean distance d between C and a truncated motif $C_{\alpha,\text{trunc}}, C_\alpha \in \Xi_\alpha$, a vector comprised of all elements of a motif except for the last one, $C_{\alpha,\text{trunc}} = (\eta_1^\alpha, \ldots, \eta_{L-1}^\alpha)$ for $C_{\alpha,\text{trunc}} = (\eta_1^\alpha, \ldots, \eta_L^\alpha)$. If $d < \varepsilon$ (where ε is a parameter of the algorithm) then the last element of the motif C_α becomes a possible predicted value at the position $t + h$. Thus, the set of possible predicted values at position $t + h$ for pattern α is defined as $\widehat{S}_{t+h}^{p,\alpha} = \{\eta_{L-1}^\alpha : \rho(C_{\alpha,\text{trunc}}) \le \varepsilon\}$.

In turn, a set of sets of possible predicted values corresponding to all patterns $\widehat{S}_{t+h}^{(p)}$ becomes a union of sets $\widehat{S}_{t+h}^{(p,\alpha)}$ and can be defined as $\widehat{S}_{t+h}^{(p)} = \cup_{\alpha\in\aleph}\widehat{S}_{t+h}^{(p,\alpha)}$. Finally, unified prediction value $\widehat{y}_{t+h}$ is calculated as a function of $\widehat{S}_{t+h}^{(p)}$. Previously predicted intermediate positions $\widehat{y}_{t+i}$ are used as inputs for the new iterations of the algorithm until a prediction horizon h is reached. Let us label each distinct iteration as a "prediction step." Performing the prediction step repeatedly produces an operator f_h: $\widehat{S}_{t+h} = f_h(\{y_t, \ldots, y_{t-s+1}\})$, which yields a set of possible predicted values for position $t + h$. The size of the set is directly proportional to the number of employed patterns. It allows for an implementation of algorithms for determining the final, unified predicted value at $t + h$ as well as identifying nonpredictable points. These algorithms are discussed later in the paper.

In addition to the set of possible prediction values $\widehat{S}_{t+h}^{(p)}$, we can calculate a set of weights $\Omega_{t+h} = \{\omega_{t+h}^i\}, i = 1..N_h$, which characterizes comparative significance of individual possible prediction values $\widehat{y}_{t+h}^i$. Alternatives presented below can be used to determine weights.

The first technique relies on the notion of a "reliability" of individual intermediate predictions, used to calculate $\widehat{y}_{t+h}^i$. Each prediction step results in a greater error of the final prediction at position $t + h$. Thus, in order to offset this error growth, we can assign a weight to an element of a sets of possible predicted values based on the number of prediction steps made in order to calculate it. We assign weights $\omega_i = 1$ to observed values (thus indicating that they are 100% reliable). Given a possible prediction value $\widehat{y}_{t+h}^i$ (either at the final point $i = t + h$ or intermediate points $t < i < t + h$), calculated based on preceding points $y_{t+h-k_{L-1}^\alpha}, y_{t+h-k_{L-1}^\alpha-k_{L-2}^\alpha}, \ldots, y_{t+h-k_{L-1}^\alpha-\ldots-k_1^\alpha}$ (predicted or observed) with corresponding weights $\omega_{t+h-k_{L-1}^\alpha-\ldots-k_1^\alpha}$ using a pattern $\alpha = (k_1^\alpha, \ldots, k_{L-1}^\alpha)$, the weight of $\widehat{y}_{t+h}^i$ is calculated as an average of weights of the preceding points of α times a step-down factor λ:

$$\omega_{t+h}^i = \lambda \frac{1}{L-1} \sum_{l=1}^{L-1} \omega_{t+h-\sum_{j=1}^{l} k_j^\alpha}. \quad (4)$$

The step-down factor λ ensures that predicted points receive a progressively smaller weight compared to the observed and earlier predicted points. We typically used $\lambda = 0.99$. We used the average weight of those possible predicted values that were used to calculate the unified predicted value in order to calculate the weight of this unified predicted value (either at the final of intermediate points).

The second technique considers ω_i to be inversely proportional to the distance between observed values and the motif chosen for the prediction:

$$\omega_{t+h}^{i} = \frac{\varepsilon - \rho\left(C, C_{\mathrm{trunc}}^{\alpha}\right)}{\varepsilon}, \tag{5}$$

where ε is a small threshold (typically equal to 0.05 for the purposes of the current simulation).

The third approach calculates ω_i as a product of the results of the previous two methods.

3.5. Unified Predicted Value. The method of calculating a unified predicted value (UPV) depends on whether the algorithm's parameter $p = 1$ or $p > 1$. If it is, the UPV is calculated based solely on the set of possible prediction values associated with the current position: $\widehat{y}_{t+h} = g(\widehat{S}_{t+h}^{(p)}) \equiv g(\widehat{S}_{t+h})$. There are several techniques that can be used to extract UPV from $\widehat{S}_{t+h}$:

(1) [avg] Averaging over $\widehat{S}_{t+h}$: $\widehat{y}_{t+h} = 1/N_{t+h}\sum_{i=1}^{N_{t+h}}\widehat{y}_{t+h}^{i}$.

(2) [wavg] Taking a weighted average of $\widehat{S}_{t+h}$: $\widehat{y}_{t+h} = 1/\sum_{j=1}^{N_{t+h}}\omega_j\sum_{i=1}^{N_{t+h}}\omega_i\widehat{y}_{t+h}^{i}$.

(3) [clc] Clustering $\widehat{S}_{t+h}$ using the DBSCAN algorithm and selecting center of the largest cluster Q_{j^*}: $\widehat{S}_{t+h} = \cup_j Q_j$, $\quad Q_i \cap Q_j = \varnothing, q_j = |Q_j|, j^* = \operatorname{argmax}_j(q_j)$. DBSCAN [35] demonstrates good performance for one-dimensional data and was deemed acceptable for the task. The exact value of the UPV in this case is $\widehat{y}_{t+h} = 1/|Q_{j^*}|\sum_{\widehat{y}_{t+h}^{(i)}\in Q_{j^*}}\widehat{y}_{t+h}^{(i)}$.

(4) Clustering only elements of $\widehat{S}_{t+h}$ with weights exceeding some threshold ω_0.

(5) Two previous techniques can be applied to fuzzy sets. Normalized weights can be viewed as values of a membership function that indicates belonging to a set of possible predicted values.

(6) Clustering $\widehat{S}_{t+h}$ and picking the center of a randomly chosen cluster based on the sum of weights of its elements relative to the sum of weights of all possible prediction values (roulette wheel).

(7) [mf] Choosing the mode of $\widehat{S}_{t+h}$.

(8) [mfp] Choosing the mode of $\widehat{S}_{t+h}$ and adding a small amount of uniformly distributed noise to it: $\widehat{y}_{t+h} = 1/|Q_{j^*}|\sum_{\widehat{y}_{t+h}^{(i)}\in Q_{j^*}}\widehat{y}_{t+h}^{(i)} + \zeta(\Delta)$, where $\zeta(\Delta)$ is a normally distributed random variable with zero mean and variance $\Delta \geq 0$.

In the case when $p > 1$, we introduce a concept of a "prediction trajectory" (a sequence of possible prediction values for every position from t to $t+h$). Let us denote an s-th possible prediction trajectory (PPT) as $\widehat{\xi}_{t+h}^{(s)} = (\widehat{y}_{t+1}^{(s)}, \ldots, \widehat{y}_{t+h}^{(s)})$ with $\widehat{\xi}_{t+h}^{(s)}(i) = \widehat{y}_{t+i}^{(s)}$ being its value at i-th position and $S_{\max}$ being the maximum number of trajectories. Then the set of all PPTs ending at position $t+h$ would be defined as $\widehat{\Xi}_{t+h}$, with $\widehat{\Xi}_{t+h}(i)$ being a set of values of its trajectories at position i. Naturally, $\widehat{S}_{t+h} \equiv \widehat{\Xi}_{t+h}(h)$ Algorithm 2.

Base prediction algorithm:

We have employed several different ways of calculating possible prediction trajectories:

(1) [trp] Randomly perturbed trajectories, where a UPV is calculated for each intermediate position using technique number 8.

(2) Starting a new trajectory at the center of each cluster of $\widehat{S}_{t+h}$, thus making the total number of trajectories increase exponentially with the number of steps (a garden of forking trajectories).

(3) Starting a new trajectory at the center of each cluster of $\widehat{S}_{t+h}$ and assigning a weight to each trajectory as a product of the weights of its individual points; trajectories with weights lower than a certain threshold are filtered out at each step of the iteration and do not form new trajectories.

Regardless of the technique used, the UPV is calculated as an average of the last points of all trajectories from the set $\widehat{\Xi}_{t+h}$: $\widehat{y}_{t+h} = 1/S_{\max}\sum_{j=1}^{S_{\max}}\widehat{\xi}_{t+h}^{j}(h)$. Based on the results of the research, the first technique proved to be the most efficient one.

3.6. Quality Measures of Identification of Nonpredictable Points. Determining whether a point is predictable or not relies on either its set of possible prediction values or its set of prediction trajectories. For each prediction horizon h, we may consider the following set:

$$\mathrm{NP}(t,m,h) = \left\{(t+i\colon\ t+m \leq t+i \leq t+h)\ \&\left(y_{t+i} \in Y_2 \& \zeta\left(\widehat{S}_{t+i}^{(p)}\right) = 0\right)\right\}. \tag{6}$$

A set of nonpredictable points at intermediate positions is defined as $\mathrm{NIP}(t,h) = \mathrm{NP}(t,1,h)$. A full set of all nonpredictable points from the test set is defined as follows:

$$\mathrm{NP}(h) = \cup_{t+h\colon\ y_{t+h}\in Y_2}\mathrm{NP}(t,h,h) \equiv \left\{t+h\colon\ y_{t+h} \in Y_2 \& \zeta\left(\widehat{S}_{t+i}^{p}\right) = 0\right\}. \tag{7}$$

```
procedure Base prediction ($(Y_1, h)$)
    $\varepsilon \longleftarrow 0.01$
    normalize observations $Y_1$
    $t \leftarrow$ index of last known observation
    for $i \longleftarrow 1..h$ do
        for $\alpha \epsilon \aleph(L, k_{\max})$ do, $\alpha = (k_1^{(\alpha)}, \ldots, k_{L-1}^{(\alpha)})$
            for $l \longleftarrow 1..L$ do
                $\eta_l^{(\alpha)} \longleftarrow y_{t-k_{L-1}^{(\alpha)}-\ldots-k_{L-i+1}^{(\alpha)}}$
            end for
            $C_\alpha \longleftarrow (\eta_l^{(\alpha)}, \ldots, \eta_L^{(\alpha)})$
            $C \longleftarrow (y_{t+h-k_{L-1}^{(\alpha)}-\ldots-k_1^{(\alpha)}}, y_{t+h-k_{L-1}^{(\alpha)}-\ldots-k_2^{(\alpha)}}, \ldots, y_{t+h-k_{L-1}^{(\alpha)}})$
            $\mathrm{Trunc}(C_\alpha) \longleftarrow (\eta_l^{(\alpha)}, \ldots, \eta_L^{(\alpha)})$
            if $\rho(C, \mathrm{Trunc}(C_\alpha)) < \varepsilon$ then, $\rho$—Euclidian distance
                add $\eta_L^{(\alpha)}$ to $\widehat{S}_{t+i}$
            end if
        end for
        if point $t + i$ is predictable then
            calculate $\widehat{y}_{t+h}$ using corresponding algorithm
        end if
    end for
end procedure
```

Algorithm 2: The prediction procedure.

In other words, this is a set of all positions that the algorithm failed to produce a prediction value for. It should be said that a point becomes nonpredictable if any of the following cases are true:

(1) Its set of possible predicted values is empty

(2) It is impossible to calculate a unified prediction value based on its set of possible predicted values

These sets rely heavily on the specific one-step-ahead prediction algorithm used, the value of, and the technique of identifying nonpredictable points. In order to develop evaluation criteria for algorithms identifying nonpredictable points, we need to consider two extreme cases:

(1) Not identifying nonpredictable points at all (the algorithm always uses the closest available motif);

(2) Assuming that the algorithm possesses *a priori* information about the actual values at intermediate positions. The point is marked as nonpredictable if the difference between the predicted and actual values $d \geq \varepsilon$. It is worth noting that the algorithm does not replace its predictions with the true values: it simply excludes clearly erroneous predictions from the following prediction operations instead. In our internal team's slang, this algorithm is named "the daemon", Socrates is known to have had his own daemon, who gave him advice in his most painful situations.

Remaining points comprise a set of ground-truth nonpredictable points for a given algorithm. We can create a set

$$\mathrm{GTNP}(t, m, h) = \{t + i\colon\ (t + m \leq t + i \leq t + h) \& (y_{t+i} \in Y_2) \& (\rho(y_{t+i}, \widehat{y}_{t+i}) \geq \varepsilon)\}, \tag{8}$$

for each point $t + h$ that we are predicting, where $\widehat{y}_{t+i} = g(\widehat{S}_{t+i}^{(p)})$ is the unified prediction value of the set of possible prediction values $\widehat{S}_{t+i}^{(p)}$. Consequently, the set of positions that the algorithm failed to make a prediction for is defined as follows:

$$\begin{aligned} \mathrm{GTNP}(t, h) &= \mathrm{GTMP}(t, 1, h) \\ &= \{t + i\colon\ (t + 1 \leq t + i \leq t + h) \& (y_{t+i} \in Y_2) \& (\rho(y_{t+i}, \widehat{y}_{t+i}) \geq \varepsilon)\}. \end{aligned} \tag{9}$$

Also, the set of nonpredictable points as follows:

$$\mathrm{GTNIP}(t,m,h)=\{t+i:\ (t+m\le t+i\le t+h)\&(y_{t+i}\in Y_2)\&(\rho(y_{t+i},\hat{y}_{t+i})\ge\varepsilon)\}. \tag{10}$$

Then, the set of ground-truth nonpredictable points becomes

$$\mathrm{GTNP}(t,m,h)=\cup_{t+h:y_{t+h}\in Y_2}\mathrm{GTNP}(t,h,h)\equiv\{t+h:\ (y_{t+h}\in Y_2)\&(\rho(y_{t+i},\hat{y}_{t+i})\ge\varepsilon)\}. \tag{11}$$

Simulations reveal that the second case is almost completely independent of the threshold value ε. Naturally, "real" algorithms do not possess *a priori* information about actual values at intermediate positions y_{t+i}, since they deal only with either sets of possible predicted values $\hat{S}^{(p)}_{t+i}$ or sets of predicted trajectories $\hat{\Xi}_{t+i}$. Nevertheless, this method of identifying nonpredictable points is useful for determining a lower boundary of the prediction error. Methods of identifying nonpredictable points should be developed in such a way as to approximate this boundary as closely as possible. In the team's internal slang, we referred to them as "approximating the daemon."

Graphs in the Figure 3 exemplify dependences of the number of nonpredictable points (Figure 3(a)) and the average error on predictable points (Figure 3(b)) on the prediction horizon h for the two extreme cases. The blue curves correspond to the first extreme case (intermediate nonpredictable points are not identified at all), and orange, to the second (intermediate nonpredictable points are identified with *a priori* information). It can be seen that in the first case the number of nonpredictable points is equal to zero and the prediction error grows exponentially with h. The second extreme algorithm demonstrates the opposite: the number of nonpredictable points grows exponentially with h, while the prediction error function remains nearly constant, bounded, and rather small. It is obvious that the first algorithm minimizes the functional (3), neglecting the functional (4); the second one minimizes the functional (4), neglecting the functional (3).

Figures 4(a) and 4(b) display the true values (blue solid) and intermediate predicted values (red dashed) for the Lorenz series for the first (Figure 4(a)) and second (Figure 4(b)) extreme cases. From the latter subfigure, we notice that the second algorithm does not make a prediction for all points, while making rather accurate predictions where it "decides" to predict. 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 Figure 4(a)) that is identified as nonpredictable by the second algorithm, the first algorithm must predict at this point (as with any point), and this immediately ruins the MSA prediction process, causing the predicted trajectory to diverge from the true one.

The results of algorithms identifying nonpredictable points fall somewhere between these two extreme cases. In the context of multi-step-ahead prediction, the second case presents more interest since it allows one to make a prediction up to a fairly large number of steps ahead. From this point, all algorithms discussed are attempt to follow it.

A large-scale simulation suggests that a forecast algorithm will be efficient only if its set of identified nonpredictable points is sufficiently close to the set of ground-truth nonpredictable points $\mathrm{GTNP}(h)$. Difference between the two constitutes an auxiliary quality measure for the techniques of identifying nonpredictable points:

$$I_3=\left|\frac{\mathrm{GTNP}(h)}{\mathrm{NP}(h)}\right|\cup\left|\frac{\mathrm{NP}(h)}{\mathrm{GTNP}(h)}\right|. \tag{12}$$

Two possible opposite scenarios can be observed: either the difference $|\mathrm{GTNP}(h)/\mathrm{NP}(h)|$ is large whereas the difference $|\mathrm{NP}(h)/\mathrm{GTNP}(h)|$ is small, or vice versa. The first case corresponds to an overly optimistic algorithm, whereas the second, to an overly conservative one.

4. Identification of Nonpredictable Points

Before we begin talking about different ways of identifying nonpredictable points, we need to establish a concept of a "perfectly predictable" point. A "perfectly predictable" point is the one which set of possible prediction values consists of a single cluster containing the vast majority of its possible predicted values. On the other hand, sets of possible prediction values for nonpredictable points tend to either consist of multiple equally sized clusters or be nonclusterable at all. In terms of probability distributions, predictable points correlate with a unimodal symmetric distribution with a relatively small variance, whereas nonpredictable ones correlate either with a uniform distribution or with a distribution with several pronounced modes.

In order to characterize nonpredictable points, we employed the following quantities:

(1) Multimodality: a percent of possible predicted values remote from the main mode

(2) Difference between α and $1-\alpha$ percentiles

(3) Second, third, and fourth central moments, as well as kurtosis of the distribution and its entropy

We performed a large-scale simulation on the validation set Y_3 by calculating sets of nonpredictable points $\mathrm{NP}(h)$ and the ground-truth nonpredictable points $\mathrm{GTNP}(h)$ for each feature separately. A comparison of these sets makes it possible to estimate the best threshold values for each feature. It revealed that each feature by itself does not cause the

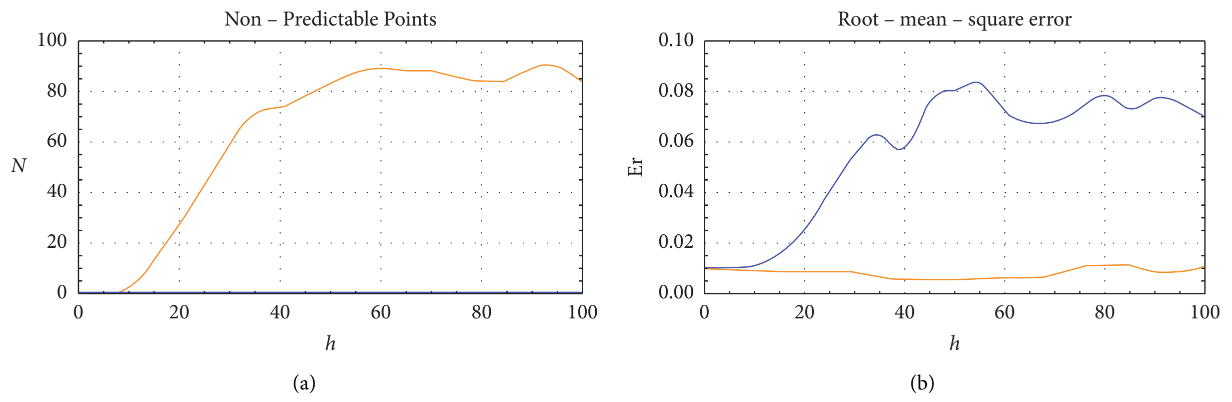

Figure 3: Lorenz series. The number of nonpredictable points (a) and RMSE (b) vs. a prediction horizon for the two extreme cases. The blue curves correspond to the first extreme case (intermediate nonpredictable points are not identified at all); orange corresponds to the second one (intermediate nonpredictable points are identified with a priori information).

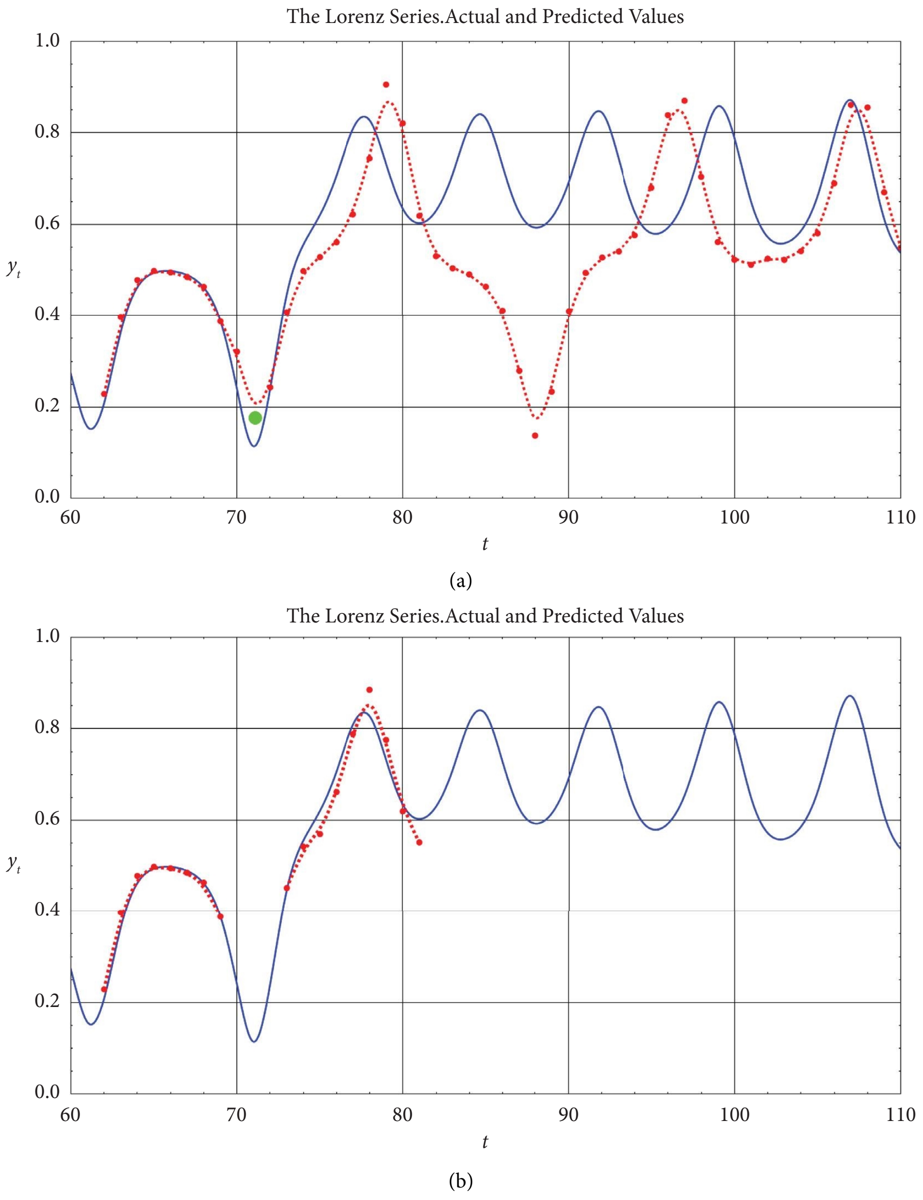

Figure 4: True (blue solid) and intermediate predicted (red dashed) values for the Lorenz series for the first (a) and second (b) extreme cases. A green dot indicates a point that the algorithm that uses *a priori* information identifies as nonpredictable.

objective to become larger; therefore, it becomes necessary to take into account multiple characteristics at once.

When clustering sets of possible prediction values, we take into consideration the following points:

(1) Total number of clusters

(2) Size of the largest cluster compared to the size of the entire set

(3) Relative difference in the size of clusters

We use the following machine learning techniques to separate the aforementioned features into regions corresponding to predictable and nonpredictable points, respectively:

(1) [lr] Logistic regression

(2) [svm] Support vector machine

(3) [dt] Decision tree

(4) [knn] K-nearest neighbors clustering

(5) [mlp] Multilayer perceptron with a single layer with 8 neurons

In order to test this approach, a new validation set Y_3 independent of both Y_1 and Y_2 needs to be introduced: $Y_1 \cap Y_3 = \varnothing, Y_2 \cap Y_3 = \varnothing$. We label Y_3 using the set of ground-truth nonpredictable points. Since the number of nonpredictable points exceeds the number of predictable ones for larger prediction horizons, a sample is balanced by SMOTE (synthetic minority oversampling technique) methods [36]. This method generates new elements in the neighbourhood of the smaller cluster.

In addition to the above classifying methods, we employ their ensembles:

(1) [adb] AdaBoost: an algorithm training each subsequent classifier on the elements incorrectly classified by the previous classifier

(2) [lrs] Stacking: applying several classifiers to produce a feature space for a combined algorithm

(3) [lrv] Voting: assigning a label to each element by the classifiers' votes

When analyzing a set of possible prediction trajectories $\widehat{\Xi}_{t+h}$ using the second approach, trajectories converging at certain points indicate predictability. Thus, predictable points correspond to the regions of convergence of the possible prediction trajectories.

Several techniques for identifying nonpredictable points were developed:

(1) Calculating an average spread over sets of possible prediction values $k_{\mathrm{avg}} = 1/|Y_3| \sum_{t+h \in Y_3} |\widehat{\Xi}_{t+h}(h) - \widehat{\Xi}_{t+h}|$. A point is classified as nonpredictable if its spread exceeds k_{avg} times some factor ϑ.

(2) Instead of considering the spread itself, one can consider its variation over a number of successive positions. The first point is classified as nonpredictable if the spread increases monotonously. Based on the results of a simulation, it appears that the best option here is to consider three successive points.

(3) For the trajectories relying on reduced initial information, we may compare the last value of a trajectory starting at position $t, \widehat{y}_{t+h} = \widehat{\xi}^{0}_{t+h}(h)$ with a weighted average of the last values of trajectories starting at positions $t-1, t-2, \ldots, t-a$: $\widetilde{y}_{t+h} = \sum_{j=1}^{a} \omega_j \widehat{\xi}^{j}_{t+h+j}(h), \omega_j = 1/2^i$. A point is classified as nonpredictable if $|\widehat{y}_{t+h} - \widetilde{y}_{t+h}| > \varepsilon$. The closer a trajectory starts to position t, the greater its weight is.

(4) Another technique designed for trajectories that rely on reduced initial information compares average modulus of the difference between the last points of trajectories starting at a position t and trajectories starting at other positions $1/a \sum_{j=1}^{a} |\widehat{\xi}^{0}_{t+h+j}(h) - \widehat{\xi}^{j}_{t+h+j}(h)|$ with ε.

(5) Similar to the previous case, we calculate a weighted average of the differences $\sum_{j=1}^{a} \omega_j |\widehat{\xi}^{0}_{t+h+j}(h) - \widehat{\xi}^{j}_{t+h+j}(h)|$ with weights $\omega_j = 2(a-j+1)/a(a+1)$.

(6) It is also possible to apply all of the listed techniques to a set of final points of $\widehat{\Xi}_{t+h}$.

Tables in the next section include results for extreme cases:

(i) [fp] (forced prediction) symbolizes the fact that nonpredictable points are not identified at all and the calculation of predicted values is forced at all intermediate points

(ii) [ab] (absent) implies that a set of nonpredictable points consists only of points that the algorithm failed to find motifs ($\widehat{S}^{p}_{t+h} = \varnothing$) for and sets of possible prediction values are not analyzed (the first extreme case)

(iii) [id] (ideal): sets of possible predicted values are constructed using *a priori* information (the second extreme case)

5. Numerical Results

Methods discussed in the previous sections are applied to the benchmark (Lorenz) and the real-world (electric grid load values) time series. Clustering is used to generate samples using all possible patterns consisting of four elements ($L = 4$) with the maximum distance between neighboring positions $K_{\max} = 10$. Thus, the total number of used patterns equals $|\aleph(L, K_{\max})| = (K_{\max})^{L-1} = 1000$.

The Lorenz series is calculated by integrating the Lorenz system with parameters $\sigma = 10, b = 8/3, r = 28$ using the Runge–Kutta fourth-order method with integration step $\Delta = 0.1$. The result is a typical chaotic series used as a conventional benchmark for testing chaotic time series forecasting methods.

In order to distinguish between chaotic and simply noisy series, we check positions of entropy and complexity on their respective planes. The values of these for the Lorenz series are 0.68 and 0.45, respectively, thus indicating the series as chaotic. Furthermore, the highest Lyapunov exponent λ

Table 1: The Lorenz series: nonpredictable points and error measures.

Cl/Trj	UPV	NP	$h = 1$			$h = 10$			$h = 50$			$h = 100$		
			NP (%)	MAPE	RMSE	NP (%)	MAPE	RMSE	NP (%)	MAPE	RMSE	NP (%)	MAPE	RMSE
cl	*avg*	*cm*	0	0.16	0.20	0	0.19	0.22	0	0.19	0.24	0	0.23	0.28
		ap	15	0.17	0.21	40	0.18	0.21	94	0.14	0.19	99	0.28	0.28
		en	36	**0.01**	0.01	49	0.01	0.02	76	0.16	0.19	81	0.26	0.32
	wavgd	*cm*	0	0.17	0.21	0	0.20	0.23	0	0.23	0.27	0	0.22	0.28
	wavgl	*cm*	0	0.16	0.20	0	0.18	0.22	0	0.20	0.24	0	0.24	0.29
	wavgc	*cm*	0	0.17	0.21	0	0.19	0.23	0	0.23	0.27	0	0.22	0.28
		lr	15	**0.01**	0.02	56	**0.02**	0.03	97	0.03	0.04	100	—	—
		svm	15	**0.01**	0.02	25	**0.02**	0.03	38	0.19	0.26	40	0.23	0.29
		dt	68	**0.01**	0.01	65	0.03	0.04	99	0.73	0.73	100	—	—
		knn	55	**0.01**	0.02	53	0.03	0.04	83	0.18	0.25	92	0.17	0.23
		mlp	21	**0.01**	0.02	39	**0.02**	0.03	75	0.15	0.21	96	0.27	0.29
		adblr	15	**0.01**	0.02	54	**0.02**	0.03	98	**0.02**	**0.02**	100	—	—
		adbsvm	21	**0.01**	0.02	41	**0.02**	**0.02**	85	0.18	0.28	97	0.18	0.22
		lrv	20	**0.01**	0.02	33	**0.02**	0.03	59	0.16	0.22	85	0.15	0.22
		lrs	24	**0.01**	0.02	33	**0.02**	0.03	58	0.18	0.25	85	**0.13**	0.17
		rg	0	0.02	0.03	44	**0.02**	0.04	69	0.12	0.17	85	0.18	0.21
		rgdbscan	0	0.02	0.03	39	**0.02**	0.04	58	0.12	0.16	59	0.20	0.25
		rgwshrt	0	0.02	0.03	11	0.03	0.04	14	0.23	0.29	11	0.20	0.25
trj	*avg*	*cm*	0	0.02	0.04	0	0.10	0.13	0	0.22	0.29	100	—	—
		ap	5	0.02	0.03	27	**0.02**	0.03	99	0.30	0.03	100	—	—
	trjp	*rgdbscan*	0	0.08	**0.01**	0	0.09	0.03	84	0.13	0.04	83	0.17	**0.04**

Bold values represent column minimums for ease of identification of the corresponding algorithms.

calculated using the Eckman algorithm for the Lorenz series has a value of 0.92, which corresponds to the results of Malinetskiy and Potapov [3]. Its strictly positive value confirms an inherently chaotic nature of the series.

Numerical error of the fourth-order Runge–Kutta method is $\varepsilon(0) = \Delta t^4 = 10^{-4}$. If the maximum precision error $\varepsilon_{\max}$ is chosen as $\varepsilon_{\max} = 10^{-1}$, then an estimated horizon of predictability would be $T \sim 1/\lambda \ln \varepsilon_{\max}/\varepsilon(0) = 7.5$, or 75 integration steps.

For testing purposes, the first 3000 observations of the series were discarded to ensure that the trajectory moves in the neighbourhood of its respective strange attractor. The sizes of the training and testing sets are 10000 and 1000 observations, respectively.

The real-world series is a series of electric grid load values in Germany from 2014-12-31 to 2016-02-20 measured in 1-hour intervals. Entropy and complexity of the series are measured at 0.499 and 0.372, respectively, and its highest Lyapunov coefficient is 0.125, which indicates the series' chaotic nature. The series was analyzed in the same way as the Lorenz series, with the results presented in Tables 1–4.

A large-scale simulation suggests that classifying only those points which lack corresponding motifs as nonpredictable results in an exponential increase in both the total number of nonpredictable points and an average prediction error. Based on Figure 2, it may be concluded that the choice of a specific technique used to calculate a unified prediction value has little effect on the rate of growth. However, the choice of a specific technique of identifying nonpredictable points seems to have a much larger impact (see Tables 5 and 6).

As a quick reminder, the unified predicted value can be calculated in the following ways (see Tables 5 and 6):

When it comes to identifying nonpredictable points, several methods from various fields of machine learning and mathematics are employed:

The results of each combination of techniques are presented in the graphs and tables. The graphs present the root mean squared error (RMSE) and the mean absolute percentage error (MAPE) for each technique alongside those of the two extreme cases (see *Quality Measures of identifying non-predictable points*) for comparison.

Presented below are the graphs of two types. The first one displays a predicted trajectory up to a certain prediction horizon alongside a "real" trajectory; the second type shows error measures (namely, root mean squared error, mean absolute percentage error and the number of nonpredictable points) for the different versions of the algorithm against different prediction horizons. Furthermore, each graph of the second type also contains plots of its respective error measure for the aforementioned two extreme cases, using *a priori* information, and not identifying nonpredictable points at all.

A large-scale simulation reveals that in the first extreme case (only points that have no corresponding motifs are considered nonpredictable) both the number of nonpredictable points and errors among the predictable ones grow exponentially with the prediction horizon.

The second extreme case (where the aforementioned algorithms are also taken into consideration when determining predictability of a point) shows an opposite situation. Figure 5 demonstrates MAPE, RMSE, and the number of nonpredictable points as a function of the prediction horizon. Orange line corresponds to the method using a priori information; gray, to the version of the "rapid growth" algorithm that uses the Wishart clustering

Table 2: The Lorenz series: sets of nonpredictable points.

Cl/Trj	UPV	NP	$h = 1$			$h = 10$			$h = 50$			$h = 100$		
			Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
		cm	1.00	0.85	0.92	1.00	0.60	0.75	1.00	0.06	0.11	1.00	0.01	0.02
	avg	*ap*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		en	0.61	0.81	0.70	0.42	0.49	0.45	0.33	0.08	0.13	—	—	—
	wavgd	*cm*	1.00	0.85	0.92	1.00	0.60	0.75	1.00	0.06	0.11	1.00	0.01	0.02
	wavgl	*cm*	1.00	0.85	0.92	1.00	0.60	0.75	1.00	0.06	0.11	1.00	0.01	0.02
	wavgc	*cm*	1.00	0.85	0.92	1.00	0.60	0.75	1.00	0.06	0.11	1.00	0.01	0.02
		lr	0.84	0.84	0.84	0.45	0.61	0.52	—	—	—	—	—	—
		svm	0.84	0.84	0.84	0.72	0.57	0.64	0.67	0.07	0.12	1.00	0.02	0.03
		dt	0.31	0.81	0.44	0.37	0.63	0.46	—	—	—	—	—	—
cl		*knn*	0.40	0.76	0.52	0.47	0.60	0.52	—	—	—	—	—	—
		mlp	0.77	0.82	0.79	0.52	0.51	0.51	0.50	0.12	0.19	—	—	—
		adblr	0.84	0.84	0.84	0.48	0.63	0.55	—	—	—	—	—	—
		adbsvm	0.77	0.82	0.79	0.50	0.51	0.50	0.17	0.07	0.10	—	—	—
		lrv	0.78	0.83	0.80	0.60	0.54	0.57	0.33	0.05	0.09	—	—	—
		lrs	0.73	0.82	0.77	0.60	0.54	0.57	0.33	0.05	0.08	—	—	—
		rg	1.00	0.85	0.92	0.58	0.63	0.60	0.50	0.10	0.16	—	—	—
		rgdbscan	1.00	0.85	0.92	0.62	0.61	0.61	0.50	0.07	0.13	—	—	—
		rgwshrt	1.00	0.85	0.92	0.92	0.62	0.74	0.83	0.06	0.11	1.00	0.01	0.02
	avg	*cm*	1.00	0.73	0.84	1.00	0.73	0.84	0.03	1.00	0.05	—	—	—
trj		*ap*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	trjp	*rgdbscan*	1.00	0.73	0.84	1.00	0.73	0.84	0.03	1.00	0.05	0.05	0.80	0.10

Table 3: Electricity load series: nonpredictable points and error measures.

Cl/Trj	UPV	NP	h = 1			h = 10			h = 50			h = 100		
			NP (%)	MAPE	RMSE	NP (%)	MAPE (%)	RMSE	NP (%)	MAPE	RMSE	NP (%)	MAPE	RMSE
cl	*avg*	*cm*	0	0.26	0.30	0	0.21	0.25	0	0.20	0.24	0	0.22	0.27
		ap	30	0.25	0.30	58	0.21	0.24	98	0.25	0.26	100	—	—
		en	38	0.26	0.33	64	0.29	0.35	65	0.25	0.31	74	0.18	0.22
	wavgd	*cm*	0	0.26	0.31	0	0.22	0.26	0	0.21	0.26	0	0.24	0.29
	wavgl	*cm*	0	0.26	0.30	0	0.21	0.25	0	0.20	0.24	0	0.21	0.26
	wavgc	*cm*	0	0.16	0.22	0	0.19	0.25	0	0.22	0.27	0	0.23	0.29
		lr	96	0.21	0.22	13	0.21	0.27	33	0.19	0.23	51	0.25	0.29
		svm	68	0.14	0.18	73	0.18	0.23	100	—	—	100	—	—
		dt	97	0.28	0.29	85	0.16	0.22	100	—	—	100	—	—
		knn	90	0.26	0.27	72	0.19	0.23	100	—	—	100	—	—
		mlp	33	0.21	0.26	31	0.20	0.25	57	0.25	0.31	70	0.27	0.31
		adblr	100	—	—	13	0.25	0.31	48	0.20	0.25	63	0.23	0.29
		adbsvm	54	0.32	0.37	45	0.26	0.32	96	0.15	0.16	100	—	—
		lrv	23	0.30	0.36	41	0.25	0.32	57	0.21	0.26	57	0.22	0.27
		lrs	79	0.28	0.33	74	0.26	0.32	100	—	—	100	—	—
		rg	0	0.25	0.31	26	0.23	0.27	32	0.20	0.25	44	0.26	0.31
		rgdbscan	0	0.25	0.31	26	0.23	0.29	28	0.21	0.26	20	0.24	0.29
		rgwshrt	0	0.25	0.31	5	0.24	0.30	14	0.19	0.24	10	0.23	0.27
trj	*avg*	*cm*	0	0.18	**0.01**	0	0.37	0.04	0	0.44	0.08	0	0.46	0.09
		ap	17	**0.09**	**0.01**	28	**0.11**	**0.01**	61	**0.11**	**0.01**	63	**0.11**	**0.01**
	trjp	*rgdbscan*	0	0.10	**0.01**	0	0.12	**0.01**	63	0.34	0.05	51	0.33	0.03

Bold values represent column minimums for ease of identification of the corresponding algorithms.

algorithm for clustering sets of possible predicted values; red—to the version of the "rapid growth" algorithm that uses DBSCAN for clustering sets of possible predicted values; blue, to the version of the "rapid growth" algorithm that averages sets of possible predicted values. It should be noted that, when comparing the results with those of others [25] that also demonstrate an ability to predict up to multiple Lyapunov times ahead, the almost-constant (non-exponential) error behavior intrinsic to some of the algorithm's variants allows for making predictions up to even more Lyapunov times ahead. However, it comes at the cost of increasing number of nonpredictable points. The work by Sangiorgio et al. [25] eventually demonstrates exponential error growth.

Of all the used machine learning algorithms, logistic regression and AdaBoost ensemble (Figure 6) stand out. Their error measures remain nearly constant with increasing prediction horizon and are comparable to those produced by the algorithm using *a priori* information for identifying nonpredictable points. However, it also demonstrates an exponential growth of the number of nonpredictable points with increasing prediction horizon at a rate larger than that of the second extreme case. The method employing a multilayer perceptron, on the contrary, demonstrates a moderate rate of growth of the number of nonpredictable points along with a rapid error growth.

When considering different techniques of identification of nonpredictable points, it should be noted that simulations reveal that the versions of the algorithm based on sets of trajectories perform better than their counterparts based on sets of possible predicted values. It can be seen from the tables that the number of nonpredictable points is close to the first extreme case (the one using *a priori* information), while the average prediction error is constrained between the two. This is a typical behavior for all methods of calculating the unified predicted value and identifying nonpredictable points. It should be noted that the main objective is still achieved, instead of increasing exponentially, average prediction error remains constant and relatively small.

Figure 7 demonstrates an example of predicted energy load series (Tables 3 and 4).

Figure 8 indicates that the algorithm is able to predict values up to (and sometimes exceeding) an estimated horizon of predictability of 75 steps (thus the title of the article).

In order to compare our algorithm with the results of other researchers, we have recreated the simulation by Prof. Kurogi and colleagues [17]. Figure 8 displays the Lorenz series (blue), prediction made by prof. Kurogi (green), and the values predicted by the algorithm (dashed red line with dots). As can be discerned from the figure, while the trajectory predicted by prof. Kurogi diverges from the actual one, the algorithm discussed in the present paper does correctly identify nonpredictable points and the predicted points approximate the true trajectory quite accurately.

In conclusion, it should be noted that the result obtained on the real-world German energy time series seems to confirm that the effect of noise (stochasticity and nonstationarity) dramatically affects the forecasting accuracy (the prediction is reliable only for the first day ahead due to observation noise and the daily periodicity that affects many real phenomena) [30, 37]. However, the current paper did not specifically concern itself with the

Table 4: Electricity load series: sets of nonpredictable points.

Cl/Trj	UPV	NP	h = 1			h = 10			h = 50			h = 100		
			Precision	Recall	*F*-measure	Precision	Recall	*F*-measure	Precision	Recall	*F*-measure	Precision	Recall	*F*-measure
cl		*cm*	1.00	0.70	0.82	1.00	0.42	0.59	1.00	0.02	0.04	—	—	—
	avg	*ap*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		en	0.61	0.69	0.65	0.27	0.33	0.31	0.50	0.03	0.05	—	—	—
	wavgd	*cm*	1.00	0.70	0.82	1.00	0.42	0.59	1.00	0.02	0.04	—	—	—
	wavgl	*cm*	1.00	0.70	0.82	1.00	0.42	0.59	1.00	0.02	0.04	—	—	—
	wavgc	*cm*	1.00	0.70	0.82	1.00	0.42	0.59	1.00	0.02	0.04	—	—	—
		lr	0.06	1.00	0.11	0.88	0.43	0.57	1.00	0.03	0.06	—	—	—
		svm	0.33	0.72	0.45	0.31	0.48	0.38	—	—	—	—	—	—
		dt	0.04	1.00	0.08	0.05	0.13	0.07	—	—	—	—	—	—
		knn	0.11	0.80	0.20	0.21	0.32	0.26	—	—	—	—	—	—
		mlp	0.73	0.76	0.75	0.81	0.49	0.61	0.50	0.02	0.04	—	—	—
		adblr	—	—	—	0.93	0.45	0.61	1.00	0.04	0.07	—	—	—
		adbsvm	0.57	0.87	0.69	0.71	0.55	0.62	—	—	—	—	—	—
		lrv	0.79	0.71	0.75	0.67	0.48	0.55	1.00	0.05	0.09	—	—	—
		lrs	0.24	0.81	0.37	0.36	0.58	0.44	—	—	—	—	—	—
		rg	1.00	0.70	0.82	0.71	0.41	0.52	1.00	0.03	0.06	—	—	—
		rgdbscan	1.00	0.70	0.82	0.74	0.42	0.53	1.00	0.03	0.05	—	—	—
		rgwshrt	1.00	0.70	0.82	0.93	0.41	0.57	0.50	0.01	0.02	—	—	—
trj	*avg*	*cm*	1.00	0.83	0.91	1.00	0.72	0.84	1.00	0.39	0.65	1.00	0.37	0.54
		ap	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	trjp	*rgdbscan*	1.00	0.83	0.91	0.82	0.70	0.76	0.05	0.14	0.08	0.11	0.67	0.19

Table 5: Abbreviations for the methods to calculate the unified prediction value.

avg	Average. Averaging the set of possible prediction values or trajectories
wavgl	Weighted average, length. Averaging the set of possible prediction values or trajectories with weights being determined by a prediction length
wavgd	Weighted average, distance. Averaging the set of possible prediction values or trajectories with weights being determined by a distance to the cluster center
wavgc	Weighted average, combined. Averaging the set of possible forecast values or trajectories with weights being a product of the two previous approaches
clc	Center of the largest cluster. Choosing the center of the largest cluster of the set of possible predicted values
mf	Most frequent. Choosing the most frequently observed value in the set of possible predicted values
mfp	Most frequent, perturbed. Choosing a randomly perturbed most frequently observed value in the set of possible predicted values
trjp	Trajectory, perturbed. Perturbed average of the last points of a predicted trajectory

Table 6: Abbreviations for the methods to identify nonpredictable points.

fp	Forced prediction. Forcing the algorithm to make a prediction at each intermediate point, ignoring nonpredictable points
cm	Close motifs. Analyzing the number of "close motifs" of each intermediate point
en	Entropy. Analyzing the entropy of the sets of possible predicted values of each intermediate point
ap	A priori. Using *a priori* information
lr	Logistic regression. Using a logistic regression classifier
svm	Support vector machine. Using an SVM classifier employing polynomial kernel functions
dt	Decision tree. Applying a decision tree employing an entropy criterion without any restriction on the tree depth to the features of the sets of possible predicted values
knn	*K*-nearest-neighbors. Applying a *k*-nearest-neighbors classifier is applied to the features of the sets of possible predicted values; *k*-nearest-neighbors value is equal to 3
mlp	Multilayer perceptron. Applying a multilayer perceptron containing 8 hidden layers and utilizes a sigmoidal activation function to the features of the sets of possible predicted values
adb	AdaBoost. Assembling logistic regression classifiers using AdaBoost
adblr	AdaBoost + logistic regression. Logistic regression classifiers are assembled with AdaBoost
adbsvm	AdaBoost + SVM. Assembling SVM classifiers using AdaBoost
lrs	Logistic regression, stacking. Stacking of logistic regression classifiers
lrv	Logistic regression, voting. Assembling logistic regression classifiers by voting
dbscan	DBSCAN. Clustering the sets of possible predicted values at each intermediate point using DBSCAN and calculating the growth rate of the total number of clusters over several consecutive points
wshrt	Wishart. Clustering the sets of possible predicted values at each intermediate point using Wishart clustering and calculating the growth rate of the total number of clusters over several consecutive points
cwat	Compare weighted average, trajectories. Comparing the main predicted trajectory with a weighted average of other predicted trajectories
amd	Average modulus of difference. Calculating an average of modulus of a difference
wamd	Weighted average modulus of difference. Calculating a weighted average of modulus of a difference
wamdt	Weighted average modulus of difference, trajectory. Comparing a weighted average of modulus of a difference of the main trajectory to that of other trajectories
rg	Rapid growth. The spread grows monotonically over several consecutive points
rgdbscan	Rapid growth, DBSCAN. DBSCAN is used to obtain centers of clusters of possible predicted values
rgwshrt	Rapid growth, Wishart clustering. Wishart clustering is used to obtain centers of clusters of possible predicted values

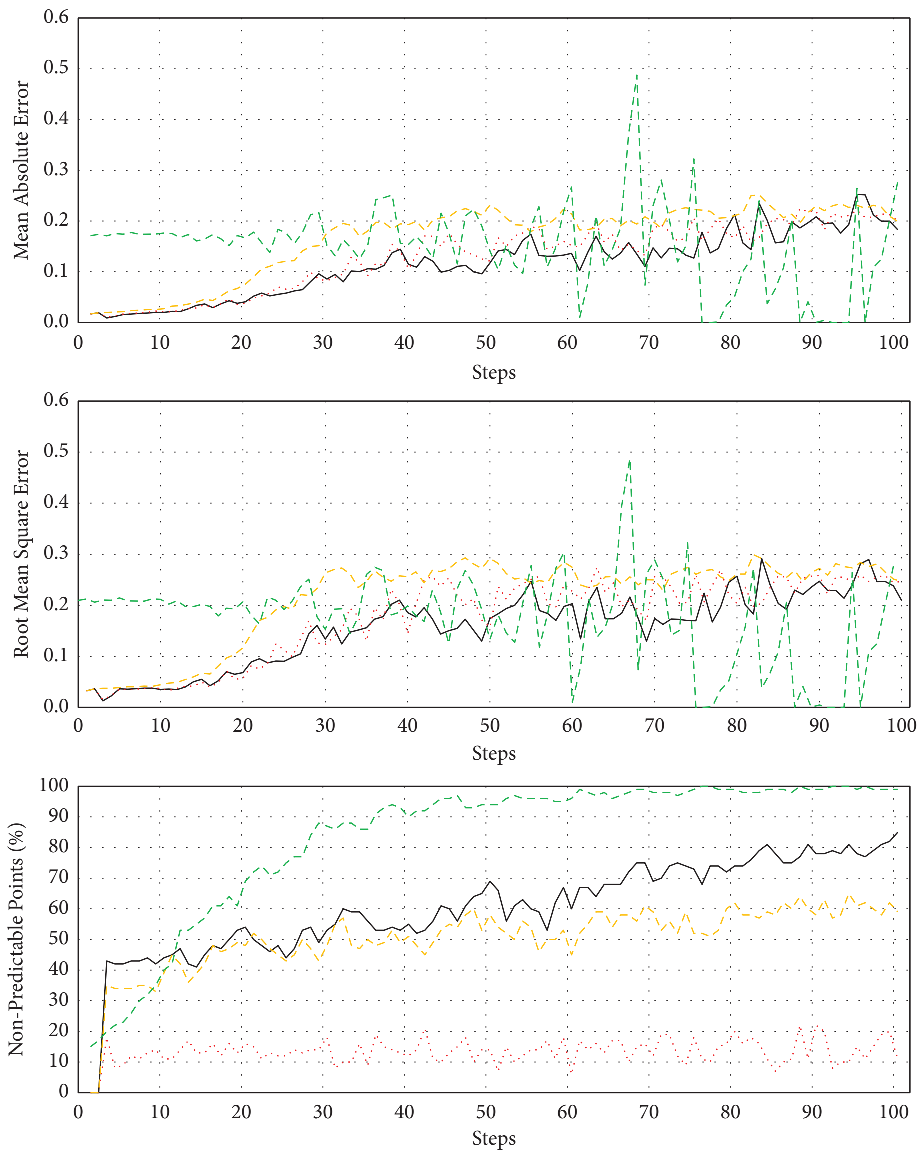

Figure 5: RMSE, MAE, and percentage of nonpredictable points of the different versions of the rapid growth algorithm compared to those of the "ideal" version. Black solid lines correspond to the basic version of the rapid growth algorithm; round red dot lines correspond to the version of the rapid growth algorithm that uses DBSCAN to cluster sets of possible predicted values; square orange dot lines correspond to the version of the rapid growth algorithm that uses Wishart clustering algorithm; green dash lines correspond to the "ideal" version that uses *a priori* information.

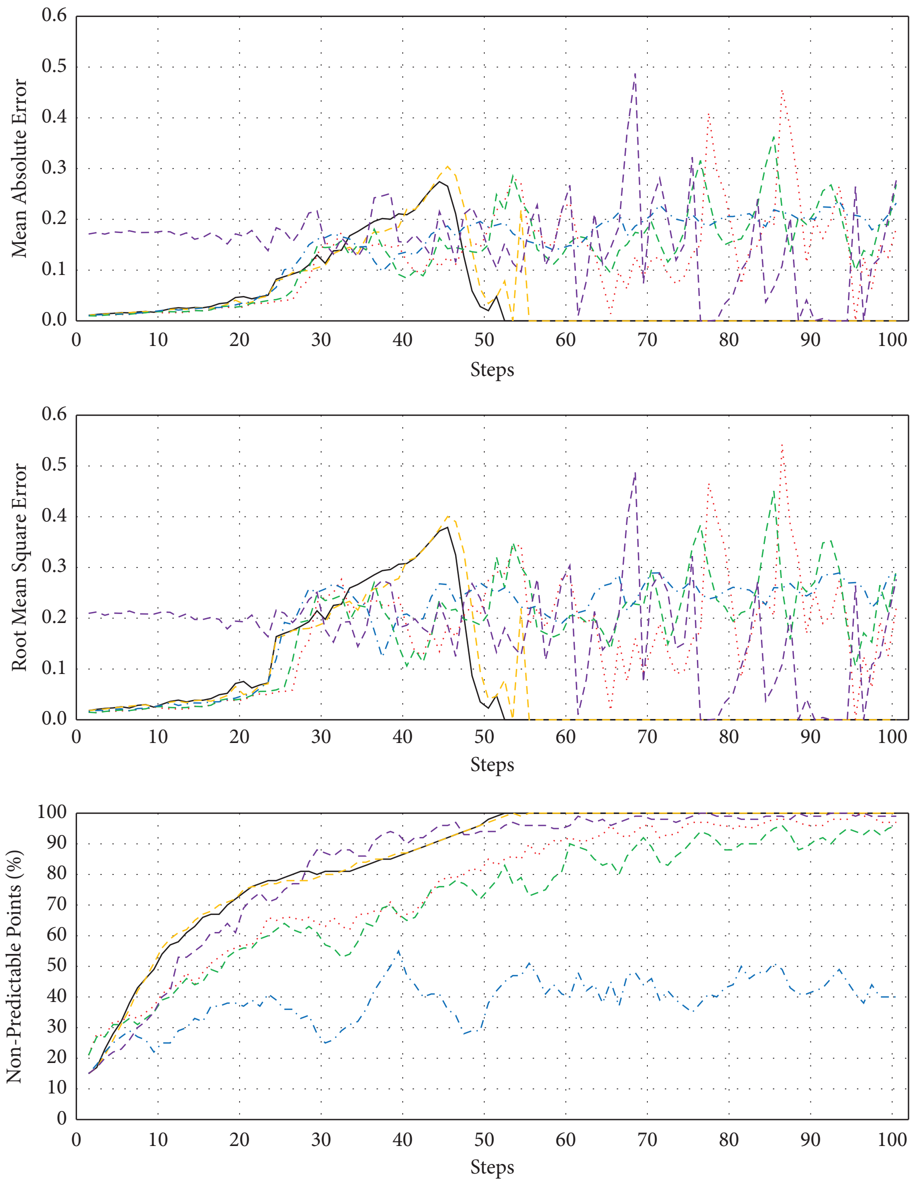

Figure 6: RMSE, MAE, and percentage of nonpredictable points of various ML algorithms compared to those of the "ideal" version. Black solid lines correspond to logistic regression with AdaBoost; round red dot lines correspond to a support vector machine with AdaBoost; square orange dot lines-to a regular logistic regression; green dash lines-to a multilayer perceptron; blue dash dot lines correspond to a support vector machine; long red dash line corresponds to the "ideal" version that uses *a priori* information. The subplot order is the same as in Figure 5.

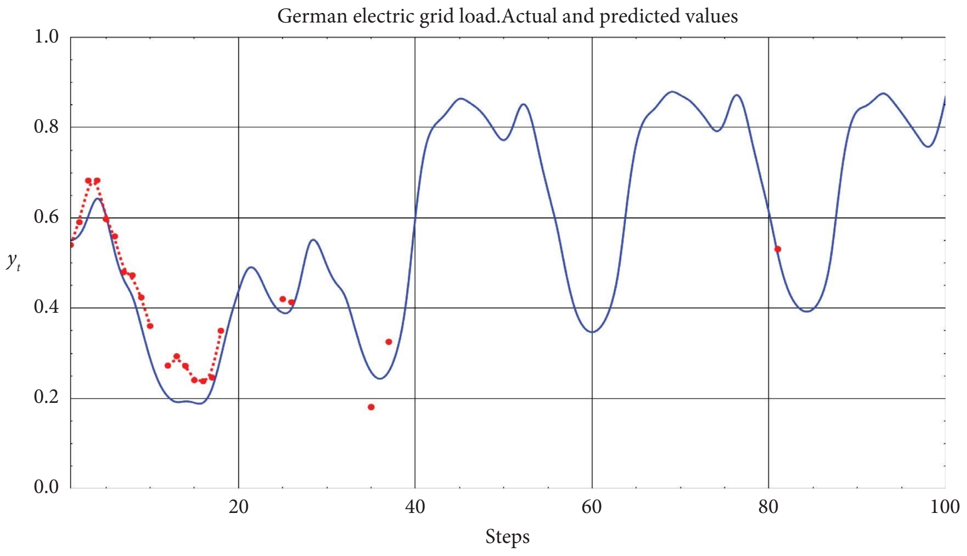

FIGURE 7: German energy time series: true trajectory (blue curve) and predicted points (red dashed curve with disks). The algorithm employs possible predicted trajectories and identifies nonpredictable points using spread growth.

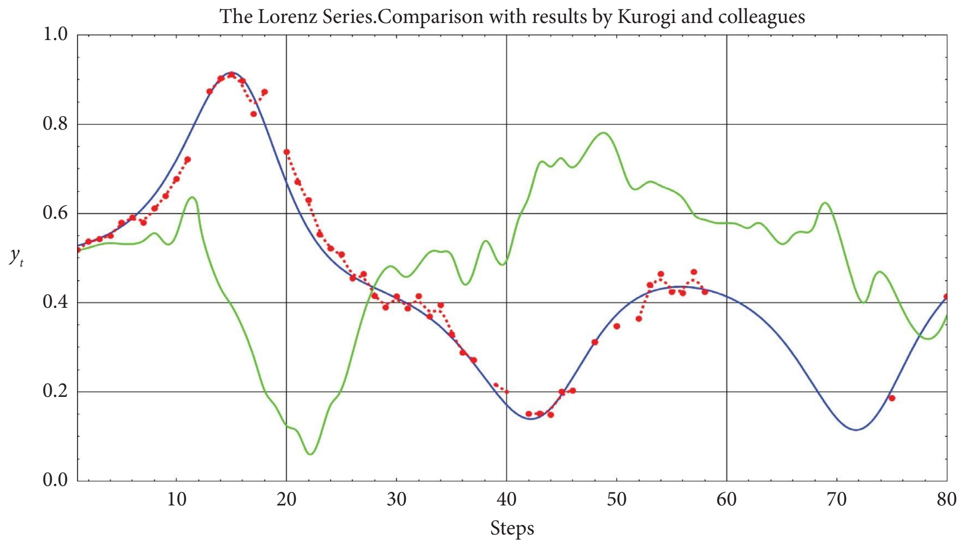

FIGURE 8: The Lorenz series: true trajectory (blue curve) and predicted points (red dashed curve with dots). A green line presents results by Kurogi and colleagues. For the sake of comparison with other literature works, note that 72 steps roughly correspond to 7 Lyapunov times.

algorithm's behavior in case of chaotic series with a significant noise component. We hope to make it the subject of future research.

6. Conclusion

Current paper introduces several novel strategies for multistep prediction of chaotic time series. Generalized z-vectors, comprised of nonsuccessive time series observations, allow for obtaining a set of possible values for each point that a prediction is being made for. Upon examining such a set, the point can be defined as either predictable (in which case its value is estimated) or nonpredictable.

The concept of nonpredictable points renders the multistep prediction as a two-objective problem, with both the number of nonpredictable points as well as the average prediction error for the predictable ones having to be minimized.

The results indicate that while the number of nonpredictable points grows exponentially with the number of steps, up to 80–90%, the average prediction error remains constant. This allows for making predictions up to a considerable number of steps ahead (although skipping some intermediate points) sometimes even exceeding the horizon of predictability. It was tested for the Lorenz and real-world time series.

It was concluded that the choice of a technique of identifying nonpredictable points has a much greater impact on the accuracy of the final prediction than the choice of a technique of estimating the actual value of the point itself.

Large-scale simulation reveals that prediction methods based on sets of possible trajectories deliver more accurate results than those based on sets of possible values, allowing for making predictions beyond the horizon of predictability.

Data Availability

All the data used to support the findings of the study are included within the article.

Disclosure

This paper is already published in the preprint given in the below link: "https://www.researchgate.net/publication/347239842_Prediction_After_a_Horizon_of_Predictability_Non-Predictable_Points_and_Partial_Multi-Step_Prediction_for_Chaotic_Time_Series" [38].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors are deeply indebted to Mr. Joel Cumberland for proofreading and editing. The results of the paper were partially presented in Chaotic Modeling and Simulation International Web Conference (CMSIM-2020, 22–24 October 2020). We are indebted to the organizers and participants of the conference. The publication was supported by the grant for research centers in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University No. 70-2021-00139. This research was supported through the computational resources of HPC facilities at the HSE University.

References

[1] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering – a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

[2] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge University Press, Cambridge, UK, 2003.

[3] G. G. Malinetskiy and A. B. Potapov, *Modern Problems of Non-linear Dynamics*, Editorial URSS, Moscow, Russia, 2002.

[4] B. P. Bezruchko and D. A. Smirnov, *Extracting Knowledge from Time Series: An Introduction to Nonlinear Empirical Modeling*, Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, Berlin, Germany, 2010.

[5] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," 1998, https://arxiv.org/abs/cs/0011032.

[6] V. A. Gromov and E. A. Borisenko, "Chaotic time series prediction & clustering methods," *Neural Computing & Applications*, vol. 2, pp. 307–315, 2015.

[7] V. A. Gromov, "Chaotic time series prediction: run for the horizon," in *Proceedings of International Conference on Software Testing, Machine Learning and Complex Process Analysis*, Tbilisi, Georgia, November 2019.

[8] S. Ben Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting," *Neurocomputing*, vol. 73, no. 10-12, pp. 1950–1957, 2010.

[9] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482–493, 2014.

[10] M. Sangiorgio and F. Dercole, "Robustness of LSTM neural networks for multi-step forecasting of chaotic time series," *Chaos, Solitons & Fractals*, vol. 139, Article ID 110045, 2020.

[11] R. Ye and Q. Dai, "MultiTL-KELM: a multi-task learning algorithm for multi-step-ahead time series prediction," *Applied Soft Computing*, vol. 79, pp. 227–253, 2019.

[12] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction," *Neurocomputing*, vol. 243, pp. 21–34, 2017.

[13] F. Martinez-Alvarez, A. Troncoso, and J. C. Riquelme, "Energy time series forecasting based on pattern sequence similarity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, pp. 1230–1243, 2011.

[14] V. A. Gromov and A. N. Shulga, "Chaotic time series prediction with employment of ant colony optimization," *Expert Systems with Applications*, vol. 39, no. 9, pp. 8474–8478, 2012.

[15] V. A. Gromov and A. S. Konev, "Precocious identification of popular topics on Twitter with the employment of predictive clustering," *Neural Computing & Applications*, vol. 28, no. 11, pp. 3317–3322, 2017.

[16] V. A. Gromov, I. M. Voronin, V. R. Gatylo, and E. T. Prokopalo, "Active cluster replacement algorithm as a tool to assess bifurcation early-warning signs for von Karman equations," *Artificial Intelligence Research*, vol. 6, no. 2, pp. 51–56, 2017.

[17] S. Kurogi, R. Shigematsu, and K. Ono, "Properties of direct multi-step ahead prediction of chaotic time series and out-of-bag estimate for model selection," in *Neural Information Processing*, C. L. Kiong, K. Y. Siah, K. W. Wai, A. Teoh, and K. Huang, Eds., vol. 8835pp. 421–428, 2014.

[18] W. Waheeb and R. Ghazali, "Multi-step time series forecasting using Ridge polynomial neural network with error-output feedbacks," in *Communications in Computer and Information Science*, M. W. Berry, A. H. Mohamed, and B. Y. Wah, Eds., vol. 652pp. 48–58, 2016.

[19] P. Ong and Z. Zainuddin, "Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction," *Applied Soft Computing*, vol. 80, pp. 374–386, 2019.

[20] R. K. Guntu, P. K. Yeditha, M. Rathinasamy et al., "Wavelet entropy-based evaluation of intrinsic predictability of time series," *Chaos*, vol. 30, no. 3, Article ID 033117, 2020.

[21] R. Wang, C. Peng, J. Gao, Z. Gao, and H. Jiang, "A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series," *Computational and Applied Mathematics*, vol. 39, no. 1, 2019.

[22] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[23] D. R. Cox, "Prediction by exponentially weighted moving averages and related methods," *Journal of the Royal Statistical Society: Series B*, vol. 23, no. 2, pp. 414–422, 1961.

[24] G. Bontempi, "Long term time series prediction with multi-input multi-output local learning," *Second European Symposium on Time Series Prediction*, vol. 15, 2008.

[25] M. Sangiorgio, F. Dercole, and G. Guariso, *Deep Learning in Multi-step Prediction of Chaotic Dynamics*, SpringerBriefs in Applied Sciences and Technology, Giorgio Guariso, 2022.

[26] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach," *Physical Review Letters*, vol. 120, no. 2, Article ID 024102, 2018.

[27] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[28] D. Canaday, A. Griffith, and D. J. Gauthier, "Rapid time series prediction with A hardware-based reservoir computer," *Chaos*, vol. 28, no. 12, Article ID 123119, 2018.

[29] K. Wang, K. Li, L. Zhou et al., "Multiple convolutional neural networks for multivariate time series prediction," *Neurocomputing*, vol. 360, pp. 107–119, 2019.

[30] M. Sangiorgio, F. Dercole, and G. Guariso, "Forecasting of noisy chaotic systems with deep neural networks," *Chaos, Solitons & Fractals*, vol. 153, Article ID 111570, 2021.

[31] P. R. Vlachas, J. Pathak, B. R. Hunt et al., "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," *Neural Networks*, vol. 126, pp. 191–217, 2020.

[32] D. Wishart, "Numerical classification method for deriving natural classes," *Nature*, vol. 221, no. 5175, pp. 97-98, 1969.

[33] H. H. Bock, *Automatic Classification*, Elsevier, Amsterdam, Netherlands, 1974.

[34] A. V. Lapko and S. V. Chentsov, *Nonparametric Information Processing Systems*, Nauka, Novosibirsk, Russia, 2000.

[35] C. C. Aggarval and C. K. Reddy, *Data Clustering: Algorithms and Applications*, CRC Press, Boca Raton, FL, USA, 2013.

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[37] D. Patel, D. Canaday, M. Girvan, A. Pomerance, and E. Ott, "Using machine learning to predict statistical properties of non-stationary dynamical processes: system climate, regime transitions, and the effect of stochasticity," *Chaos*, vol. 31, no. 3, Article ID 033149, 2021.

[38] V. A. Gromov and P. S. Baranov, "Prediction after a horizon of predictability," *Non-predictable Points and Partial Multi-step Prediction for Chaotic Time Series*, vol. 20, 2020.