mathematics

MDPI

Article

Compression-Based Methods of Time Series Forecasting †

Konstantin Chirikhin [1,2] and Boris Ryabko [1,2,*]

[1] Federal Research Center for Information and Computational Technologies, 630090 Novosibirsk, Russia; chirihin@gmail.com
[2] Department of Information Technologies, Novosibirsk State University, 630090 Novosibirsk, Russia
* Correspondence: boris@ryabko.net
† This paper is an extended version of our paper published in ESM'2020, Toulouse, France; ISF'2019, Thessaloniki, Greece.

Abstract: Time series forecasting is an important research topic with many practical applications. As shown earlier, the problems of lossless data compression and prediction are very similar mathematically. In this article, we propose several forecasting methods based on real-world data compressors. We consider predicting univariate and multivariate data, describe how multiple data compressors can be combined into one forecasting method with automatic selection of the best algorithm for the input data. The developed forecasting techniques are not inferior to the known ones. We also propose a way to reduce the computation time of the combined method by using the so-called time-universal codes. To test the proposed techniques, we make predictions for real-world data such as sunspot numbers and some social indicators of Novosibirsk region, Russia. The results of our computations show that the described methods find non-trivial regularities in data, and time universal codes can reduce the computation time without losing accuracy.

Keywords: time series forecasting; universal coding; data compression; artificial intelligence

Citation: Chirikhin, K.; Ryabko, B. Compression-Based Methods of Time Series Forecasting. *Mathematics* **2021**, *9*, 284. https://doi.org/10.3390/math9030284

Academic Editor: Ana M. Aguilera
Received: 18 December 2020
Accepted: 28 January 2021
Published: 31 January 2021

1. Introduction

The problem of time series forecasting is to estimate the future values of a process from a sequence of its observations. This task is important because it has many practical applications. Examples include predicting future stock prices and air temperature forecasting. Nowadays, there are many different approaches to solving this problem. Classical statistical models such as exponential smoothing and the autoregressive integrated moving average (ARIMA) model are very popular, highly accurate, and relatively easy to use. A detailed description of these methods can be found in [1,2]. Neural networks [3–5] are also widely used, especially on large datasets. However, there is no best method for all situations, and the development of new forecasting techniques remains relevant.

This work is based on an information-theoretic approach to time series forecasting. As is it was shown in [6], the problems of data compression and prediction are closely related and an asymptotically optimal method for predicting stationary stochastic processes can be based on a universal code (see also [7]). In [8] it was shown how any lossless data compression algorithm can be used to predict finite-alphabet and real-valued time series.

In this paper, we do not focus on asymptotic properties of algorithms and consider using this approach in practical situations. The main contributions of the work are summarized as follows. First, we describe how to use arbitrary data compressors for time series forecasting. Such compressors are a promising tool for forecasting because many of them are implementations of universal codes with numerous modifications to increase the level of compression in real-world situations. It is important to note that modern data compression techniques are based on several different approaches to universal coding: the Burrows-Wheeler Transform (BWT) [9], the Prediction by Partial Matching (PPM) [10] algorithm, the Lempel-Ziv family of algorithms [11,12], among others. Some algorithms [13–15]

search for a compact context-free grammar that unambiguously represents the sequence for compression, one can see this approach as a kind of artificial intelligence. In previous works, only some theoretical possibilities of using universal codes for prediction without any practical applications were described [8], or the use of one universal code for prediction was presented [7].

Secondly, we propose an adaptive approach to time series forecasting, which is useful in situations when we do not know in advance which data compressor is optimal for a given time series.

Thirdly, we describe how the proposed techniques can be using to predict multivariate data. To test the proposed techniques, we make forecasts for real-world data such as sunspot numbers and some social indicators of Novosibirsk region, Russia.

It should be noted that our approach can be complemented with well-known time series transformation and adjustment techniques. For example, in this work, we use differencing and smoothing in the computational experiments.

The rest of the paper is structured as follows. In the next section, we describe the mathematical foundations of using arbitrary data compression techniques for forecasting finite-alphabet time series. After that, we describe generalizations of the model to real-valued and multivariate cases. Then we give some examples of the practical use of the presented techniques. Next, we propose an adaptive algorithm that can significantly reduce computation time when multiple data compressors are used. Further, we use the proposed algorithm to predict sunspot numbers. At the end of the paper, we describe the limitations of the proposed approach and make some conclusions.

2. Data Compression and Prediction

2.1. Predicting Finite-Alphabet Time Series

Time series with finite alphabets are most convenient for forecasting using data compression algorithms. Suppose we have a sequence $X = x_1, x_2, \ldots, x_t$, $x_i \in A$, where A is a finite set (an alphabet), and we want to give a prediction for $x_{t+1}, x_{t+2}, \ldots, x_{t+h}$, $h \in \mathbb{Z}^+$. Denote as A^n the set of all sequences of lengths n over A and $A^* = \bigcup_{i=0}^{\infty} A^i$. A uniquely decodable data compression method (or code) φ is a set of mappings $\varphi_n : A^n \to \{0,1\}^*$, $n = 1, 2, \ldots$, such that for any sequence of words $x_1, x_2, \ldots, x_m$, $x_i \in A^n$, $m \geq 1$, the sequence $\varphi_n(x_1), \varphi_n(x_2), \ldots, \varphi_n(x_m)$ can be uniquely decoded as $x_1, x_2, \ldots, x_m$. The compressed size (in bits) of sequence α we denote as $|\varphi(\alpha)|$. We can get a probability distribution on A^n using φ by

$$\mathbb{P}_\varphi(X) = 2^{-|\varphi(X)|} / \sum_{Y \in A^t} 2^{-|\varphi(Y)|}. \tag{1}$$

A code φ is called universal if for any stationary and ergodic measure $\mathbb{P}$

$$\lim_{t\to\infty} |\varphi(x_1, x_2, \ldots, x_t)|/t = H(\mathbb{P})$$

with probability 1, and

$$\lim_{t\to\infty} \mathbb{E}(|\varphi(x_1, x_2, \ldots, x_t)|)/t = H(\mathbb{P}),$$

where $H(\mathbb{P})$ is the entropy rate [16] of $\mathbb{P}$, $\mathbb{E}(f)$ is the expected value of f. In [8] it was shown that if $\mathbb{P}$ is a stationary and ergodic stochastic process and φ is a universal code, (1) in certain sense is a nonparametric estimate of the unknown probability measure $\mathbb{P}(X)$. More precisely, the following theorem was proved.

Theorem 1. *If $\mathbb{P}$ is a stationary ergodic measure and φ is a universal code, then the following equalities hold:*

1. $\lim_{t\to\infty} \frac{1}{t}(-\log \mathbb{P}(x_1, x_2, \ldots, x_t) - (-\log \mathbb{P}_\varphi(x_1, x_2, \ldots, x_t))) = 0$ *with probability 1,*
2. $\lim_{t\to\infty} \frac{1}{t} \sum_{X\in A^t} \mathbb{P}(X) \log(\mathbb{P}(X)/\mathbb{P}_\varphi(X)) = 0,$
3. $\lim_{t\to\infty} \frac{1}{t} \sum_{X\in A^t} \mathbb{P}(X)|\mathbb{P}(X) - \mathbb{P}_\varphi(X)| = 0.$

We can use (1) to estimate the conditional probability that $x_{t+1} = y_1, x_{t+2} = y_2, \ldots, x_{t+h} = y_h$ for some $Y \in A^h$ as

$$\mathbb{P}_\varphi(Y|X) = \mathbb{P}_\varphi(x_{t+1} = y_1, x_{t+2} = y_2, \ldots, x_{t+h} = y_h | x_1, x_2, \ldots, x_t) = \frac{\mathbb{P}_\varphi(x_1, x_2, \ldots, x_t, y_1, y_2, \ldots, y_h)}{\mathbb{P}_\varphi(x_1, x_2, \ldots, x_t)} = \frac{2^{-|\varphi(x_1,x_2,\ldots,x_t,y_1,y_2,\ldots,y_h)|}}{\sum_{(z_1,z_2,\ldots,z_h)\in A^h} 2^{-|\varphi(x_1,x_2,\ldots,x_t,z_1,z_2,\ldots,z_h)|}}. \quad (2)$$

As we can see from (2), the conditional probability of $Y \in A^h$ depends on how well Y can be compressed after X (relative to any other $Z \in A^h$).

Suppose that A is a finite set of integers, $\mathbb{P}_\varphi(x_{t+j} = a|X) = \sum_{Y=(y_1,\ldots,y_{j-1},a,y_{j+1},\ldots,y_h)\in A^h} \mathbb{P}_\varphi(Y|X)$. We can give a point forecast as $\hat{x}_{t+j} = \sum_{a\in A} a\mathbb{P}_\varphi(x_{t+j} = a|X)$ (i.e., compute the mean over the marginal distribution for step j).

Example 1. *Consider the following sequence:*

$$X = 0001110001110001.$$

Suppose we want to make a forecast for its next two values using gzip [17] as φ. Assume that the alphabet A of the underlying process is $\{0, 1\}$. We need to compress every sequence of the form XZ, where $Z \in A^2$. The compression results along with the calculated conditional probabilities $\mathbb{P}_{gzip}(Z|X)$ are presented in Table 1.

Table 1. The results of compressing the sequences from Example 1 and the calculated conditional probabilities.

Sequence XZ	Gzip(XZ), Bytes	Gzip(XZ), Bits	$\approx \mathbb{P}_{\text{Gzip}}(Z\|X)$
0001110001110001**00**	37	296	0
0001110001110001**01**	37	296	0
0001110001110001**10**	36	288	0.004
0001110001110001**11**	35	280	0.996

For instance, $\mathbb{P}_{gzip}(11|0001110001110001) = 2^{-280}/(2^{-296} + 2^{-296} + 2^{-288} + 2^{-280}) \approx 0.996$ *and*

$$\mathbb{P}_{gzip}(0|0001110001110001) = \mathbb{P}_{gzip}(00|0001110001110001) + \mathbb{P}_{gzip}(01|0001110001110001) \approx 0 + 0 = 0,$$
$$\mathbb{P}_{gzip}(1|0001110001110001) = \mathbb{P}_{gzip}(10|0001110001110001) + \mathbb{P}_{gzip}(11|0001110001110001) \approx 0.004 + 0.996 = 1.$$

Suppose we have a finite set of data compression algorithms $F = \{\varphi_1, \varphi_2, \ldots, \varphi_k\}$. If we do not know which $\varphi \in F$ is the best predictor for a given series X, we can mix the conditional probability distributions yielded by each compressor from F by

$$\mathbb{P}_F(x_{t+1} = y_1, x_{t+2} = y_2, \ldots, x_{t+h} = y_h | x_1, x_2, \ldots, x_t) = \frac{\sum_{i=1}^{k} \omega_i 2^{-|\varphi_i(x_1,x_2,\ldots,x_t,y_1,y_2,\ldots,y_h)|}}{\sum_{(z_1,z_2,\ldots,z_h)\in A^h} \sum_{i=1}^{k} \omega_i 2^{-|\varphi_i(x_1,x_2,\ldots,x_t,z_1,z_2,\ldots,z_h)|}}, \tag{3}$$

where $\omega_i \geq 0$, $\sum_{i=1}^{k} \omega_i = 1$.

Note that (3) works in such a way that the highest probability gets $Y \in A^h$ such that $|\varphi_s(XY)| = \min_{Z\in A^h, \varphi\in F} |\varphi(XZ)|$ for some $\varphi_s \in F$, i.e., (3) selects the best compressor φ_s "automatically".

2.2. Predicting Real-Valued Time Series

Many time series that can be found in practice are sequences of real numbers. To predict such a series using data compression algorithms, we need to convert it to a sequence of integers (with loss of information, obviously). This process of conversion is known as quantization. Consider a sequence $X = x_1, x_2, \ldots, x_t$, where $x_i \in \mathbb{R}$. Denote its minimal and maximal elements as m and M respectively: $m = \min_{1\leq i\leq t}\{x_i\}$, $M = \max_{1\leq i\leq t}\{x_i\}$. Probably the simplest way of conversion is to split $[m; M]$ into a finite number n of disjoint numbered intervals $\{q_1, q_2, \ldots, q_n\}$ of equal length and replace each x_i with its corresponding interval number: if $x_i \in q_j$, then replace x_i with j. Later, we can perform the inverse conversion, replacing the indices, for example, with the medians of the corresponding intervals.

Now, we consider the question how to select the number of intervals n. On the one hand, if this value is too small, some important regularities may be missing in the converted series. On the other hand, if n is too large, a data compressor may not be able to capture the regularities due to noise in the data. One possible solution of this problem is to employ an approach similar to that used in (3). Let n be some positive power of 2: $l = \log_2 n$ is an integer greater than zero. We can mix the probability distributions, obtained using partitions into 2^k intervals, $1 \leq k \leq l$, with some weights. The partition yielded the smallest code length will have the greatest impact on the final result. Denote the number of interval that contains x_i in partition into 2^k intervals as $x_i^{[k]}$. Then the mixed probability distribution can be defined as

$$\mathbb{P}_\varphi(x_1^{[l]}, x_2^{[l]}, \ldots, x_t^{[l]}) = \frac{\sum_{k=1}^{l} \omega_k 2^{-|\varphi(x_1^{[k]},x_2^{[k]},\ldots,x_t^{[k]})|+t(l-k)}}{\sum_{k=1}^{l} \sum_{Z\in A_k^t} \omega_k 2^{-|\varphi(Z)|+t(l-k)}}, \tag{4}$$

where $A_k = \{0, 1, \ldots, 2^k - 1\}$ is a set of interval numbers, ω_k—non-negative weights, $\sum_{k=1}^{l} \omega_k = 1$.

2.3. Predicting Multivariate Data

In some practical situations, it is required to predict several related series. In such cases, we can try to utilize the connection between them to improve the accuracy of our forecasts. For example, air temperature and barometric pressure are related, and it might make sense to predict them together. Many univariate time series forecasting techniques have generalizations to the multivariate (or vector) case [18–21].

The approach based on data compression algorithms can be used in a multivariate setting too. Suppose we have a vector time series $\bar{X} = \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_t$, where $\bar{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$, $d < \infty$, $x_{ij} \in \mathbb{R}$. Let's find the minimal and maximal values for each coordinate: $m_j = \min\limits_{1 \le i \le t} \{x_{ij}\}$, $M_j = \max\limits_{1 \le i \le t} \{x_{ij}\}$, $1 \le j \le d$. As in the previous section, we split each interval $[m_j; M_j]$ into a finite number of intervals n and thus obtain n^d d-cubes. Then we need to number these cubes and replace each point $\bar{x}_i$ with the number of the cube it falls into. As a result, we get an integer sequence that can be predicted using the previously described methods.

2.4. Experiments

In this section, we use the proposed method to predict real-world data. Along with the point forecasts (which are the means of the corresponding distributions), we provide 95% confidence intervals for them. To estimate these intervals, we used the following strategy. Suppose that we want to make an h-step ahead prediction for time series $X = x_1, x_2, \ldots, x_t$. Let s be $\lfloor t/2 \rfloor$. For each series $X_s, X_{s+1}, \ldots, X_{t-h}$, where $X_i = x_1, x_2, \ldots, x_i$, we made an h-step forecast $\hat{X}_i^h = \hat{x}_{i+1}, \hat{x}_{i+2}, \ldots, \hat{x}_{i+h}$ and calculated the residuals for each step: $r_{ij} = x_{i+j} - \hat{x}_{i+j}$, $1 \le j \le h$. Then we computed the standard deviations of the residuals as $\sigma_j = \sqrt{\frac{1}{t-h-s+1} \sum\limits_{i=s}^{t-h} (r_{ij} - \bar{r}_j)^2}$, where $\bar{r}_j = (\sum\limits_{i=s}^{t-h} r_{ij}) / (t-h-s+1)$. The confidence interval for $\hat{x}_{t+j}$ was calculated as $[\hat{x}_{t+j} - 2\sigma_j; \hat{x}_{t+j} + 2\sigma_j]$.

Example 2. *Suppose we have a series*

$$X = 1.1, 1.2, 1.3, 1.2, 1.4$$

and we want to make a 2-step forecast. As was explained previously, we make 2-step ahead predictions for the series

$$X_2 = 1.1, 1.2$$

and

$$X_3 = 1.1, 1.2, 1.3.$$

Suppose that our forecast for X_2 *is* $1.3, 1.4$ *and our forecast for* X_3 *is* $1.4, 1.5$. *Then the standard deviations of the residuals for steps* 1 *and* 2 *are*

$$\sigma_1 = \sqrt{((0+0.1)^2 + (-0.2+0.1)^2)/2} = 0.1$$

and

$$\sigma_2 = \sqrt{((-0.2+0.15)^2 + (-0.1+0.15)^2)/2} = 0.05.$$

If our forecast for X *is* $1.3, 1.5$, *our confidence interval for the first step is* $[1.3 - 2 \cdot 0.1; 1.3 + 2 \cdot 0.1] = [1.1; 1.5]$, *for the second step is* $[1.5 - 2 \cdot 0.05; 1.5 + 2 \cdot 0.05] = [1.4; 1.6]$.

To get all the predictions presented below, we used the following set of data compression algorithms, combined using (3):

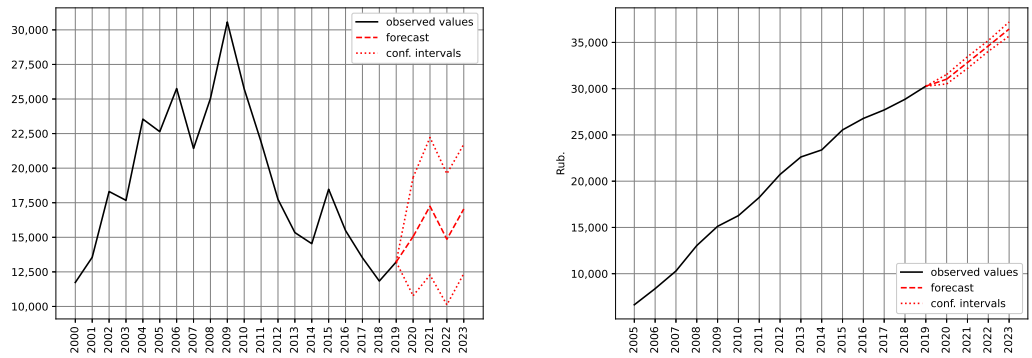
1. lcacomp (https://code.google.com/archive/p/lcacomp/)—a grammar-based data compressor proposed in [22];
2. Re-Pair (https://github.com/nicolaprezza/Re-Pair)—a grammar-based compressor, the implementation is described in [23];
3. zstd (https://github.com/facebook/zstd)—a fast lossless data compression algorithm, developed by Facebook;
4. bzip2 (https://www.sourceware.org/bzip2)—a data compressor based on the Burrows-Wheeler transform;
5. zlib (https://zlib.net/)—a data compression library that implements the DEFLATE algorithm;

6. ppmd (https://github.com/Shelwien/ppmd_sh)—an implementation of the prediction by partial matching (PPM) algorithm;
7. zpaq (https://github.com/zpaq/zpaq)—a journaling archiver optimized for user-level incremental backup of directory trees;
8. automaton (https://github.com/kchirikhin/itp)—our implementation of an algorithm based on multihead sensing finite automata [24].

We mixed these compressors with the same weights, that is, in (3) $\omega_i = 1/8$.

Several simple techniques of data transformation also were used. First, to remove trends in data we took the first difference: $y_i = x_i - x_{i-1}$. Secondly, we used smoothing: $y_i = (2x_i + x_{i-1} + x_{i-2})/4$.

Let us move on to forecasting some indicators of Novosibirsk region. In the first example, we predict the annual number of unemployed persons in the region. In Figure 1a this series along with our 4-step forecast with confidence intervals is presented. Since this series is a real-valued one, we used partitions of its range of values into 2, 4 and 8 intervals in (4). According to our forecast, in the next four years the indicator will be higher than the current level.



(**a**) (**b**)

Figure 1. (**a**) Average annual number of unemployed persons in Novosibirsk region. (**b**) Average annual monetary income of the population in Novosibirsk region.

In the second example, we predict the average annual monetary income of the population in the region. The results are shown in Figure 1b. The forecast shows that in the next 4 years the trend will continue and by 2023 the indicator will be near 35,000 rubles per year.

Our implementation is available at https://github.com/kchirikhin/itp.

3. Adaptive Method of Forecasting

As noted in Section 2.1, multiple data compression algorithms can be combined into one method of forecasting (almost) without loss of accuracy using (3). The only issue is computation time—we need to compress all sequences with every data compression algorithm that we include in our combination. In this section, we consider a way to significantly reduce computation time while maintaining near-optimal accuracy. This can be achieved using the so-called time-universal codes.

3.1. Time-Universal Codes

Suppose we want to compress a sequence $X = x_1, x_2, \ldots, x_n$, x_i belongs to some finite alphabet A. Also suppose we have a finite set of data compression algorithms $F = \{\varphi_1, \varphi_2, \ldots, \varphi_k\}$ and we want to compress X with $\varphi_s \in F$ that yields the smallest code length. The obvious way to find φ_s is to compress X with each φ_i and then to choose the best one. After that, we need to store s along with $\varphi_s(X)$ in order to be able to decompress the original data. Binary representation $< q >$ of any integer q from 0 to $k-1$ requires no more than $\lceil \log_2 k \rceil$ bits, therefore our final code word would be $< s > \varphi_s(X)$ with

length $\lceil \log_2 k \rceil + |\varphi_s(X)|$ bits. This approach works well but requires additional time proportional to k. The goal is to compress X with the best compressor using relatively small additional time.

Time-universal codes were proposed in [25] and allow making the portion of extra time asymptotically as small as desired. Let v_i be the time φ_i spends on encoding a single letter of X, $v = \max\limits_{1 \le i \le k} v_i$. Thus, an upper bound for time needed to encode X using any $\varphi \in F$ is $T = vn$. Denote as δT the amount of extra time we have to select a close to optimal data compressor from F for X, where δ is a positive constant. The total time of selection and compression becomes no more than $T + \delta T = T(1 + \delta)$. Time-adaptive and time-universal codes are defined as follows.

Definition 1. *Any method of data compression that encodes a sequence $x_1, x_2, \ldots, x_n$, $n > 0$, $x_i \in A$ by a binary word of the length $\lceil \log_2 k \rceil + |\varphi_s(x_1, x_2, \ldots, x_n)|$ for some $\varphi_s \in F$ and the time of encoding is not greater than $T(1 + \delta)$ is called as time-adaptive code and denoted as $\hat{\Phi}^{\delta}_{compr}$.*

Definition 2. *If for a time-adaptive code $\hat{\Phi}^{\delta}_{compr}$ the following equation is valid*

$$\lim_{n \to \infty} \frac{|\hat{\Phi}^{\delta}_{compr}(x_1, x_2, \ldots, x_n)|}{n} = \min_{1, \ldots, k} \lim_{n \to \infty} \frac{|\varphi_i(x_1, x_2, \ldots, x_n)|}{n},$$

this code is called time-universal.

In [25] was proposed the following simple algorithm and proved that it yields a time universal code:

1. Calculate $r = \lfloor \delta T / kv \rfloor$;
2. Find $\varphi_s \in F$ such that $|\varphi_s(x_1, x_2, \ldots, x_r)| = \min\limits_{i=1,\ldots,k} |\varphi_i(x_1, x_2, \ldots, x_r)|$, $1 \le s \le k$;
3. Compress $x_1, x_2, \ldots, x_n$ using φ_s and make the code word $< s > \varphi_s(x_1, x_2, \ldots, x_n)$.

In this work, we implemented this algorithm and used it to predict real-world time series.

3.2. Experiments

In this section, we consider sunspot numbers forecasting. Sunspots are temporary spots on the Sun with reduced surface temperature that appear darker than the surrounding areas. The sunspot number time series is provided by the WDC-SILSO (World Data Center—Sunspot Index and Long-term Solar Observations) [26]. The daily, monthly, yearly and 13-month smoothed monthly total sunspot numbers can be found on the SILSO site http://www.sidc.be/silso/datafiles. Here we used the monthly mean total sunspot number series.

We accessed the SILSO site on 20 June 2020, at that time the series had 3257 observations. The entire series is shown in Figure 2. It has an obvious cyclical component and it's known that the approximate cycle lengths is 11 years [27].

The Space Weather Services (SWS) of the Australian Bureau of Meteorology issues forecasts for the WDC-SILSO sunspot numbers and maintains an archive of this forecasts (http://listserver.ips.gov.au/pipermail/ips-ssn-predictions/), so we can compare our forecasts with them. In July 2015, SILSO adjusted the original data and SWS moved to the new version in the forecasts in February 2016. We started making predictions from that date, thus after each element with the number greater than 3205 (which corresponds to January 2016), before adding it to the end of the series, we made a 4-step forecast.

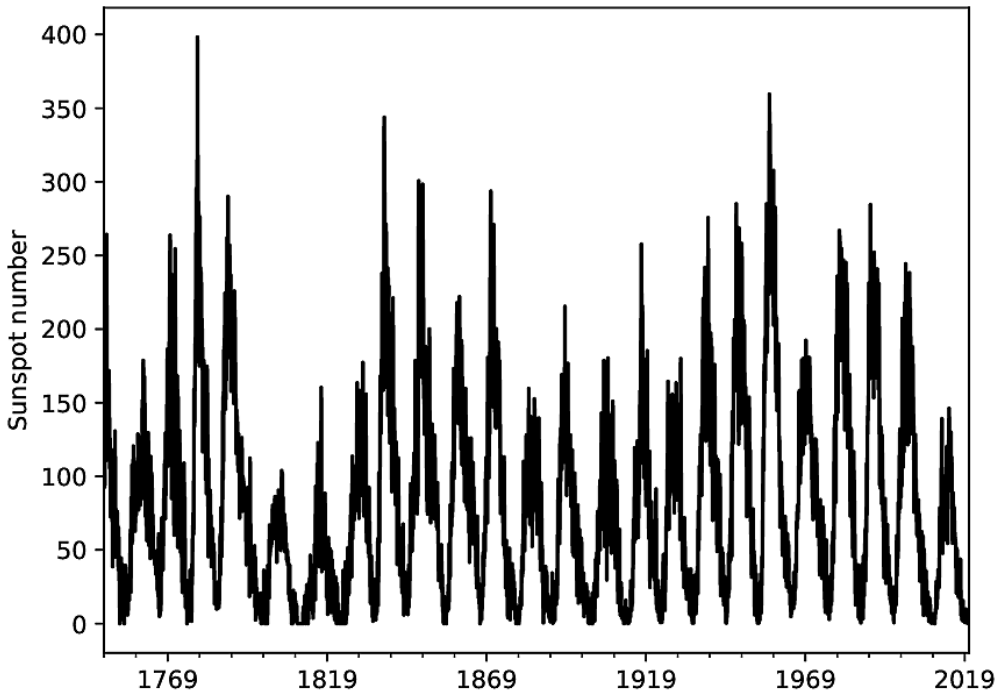


Figure 2. Monthly mean sunspot numbers.

In order to select a (close to) optimal compressor for this series, we tried to use from 10% to 100% of its values with step 10%. For instance, when 10% of the values were used, δ was $8\times$ 0.1 = 0.8 since we had 8 algorithms. In Section 2.4, the maximal number of intervals we considered when quantizing was 8; in this section, we increased this value to 16. In everything else, we used the same methodology as in Section 2.4.

For brevity, let us denote the combination of all 8 algorithms, obtained using (3), as joint method.Code lengths for different compressors, obtained by compressing the entire series (100% of values), are presented in Table 2. We can see that by code length zstd is the best compressor for this series and hence it will have the greatest contribution to the result of the joint method. For 10% trough 40% of the series values the best compressor by code lengths is ppmd, then for 50–100% zstd becomes the best compressor. The code length for ppmd and zstd are shown in Table 3.

Table 2. Compressed sizes of the sunspot number series (the minimum values obtained using partitions into 2, 4, 8 and 16 intervals).

Data Compressor	Code Length (Bits)	Data Compressor	Code Length (Bits)
zstd	8400	rp	10,488
ppmd	8528	zpaq	10,808
bzip2	9504	lcacomp	16,352
zlib	9864	automat.	63,120

To assess the accuracy of our forecasts, we calculated the mean absolute error (MAE), defined as the mean of the absolute values of the residuals, for each step 1–4. The MAEs of ppmd's and zstd's forecasts are presented in Table 4. We can see that ppmd was a bit more accurate than zstd. We tried to add more values for prediction (from 2800 to 3205), and in that case zstd was slightly more accurate for steps 1–3 (the mean absolute errors are presented in Table 5). For step 4, ppmd was more accurate.

Table 3. Compressed sizes of 10–100% of the sunspot number series values for zstd and ppmd (the minimum values obtained using partitions into 2, 4, 8 and 16 intervals).

Data Compressor	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
zstd	1248	2160	2992	3656	4416	5216	6024	6840	7672	8400
ppmd	1104	1944	2664	3592	4488	5296	6088	6952	7840	8528

We ran the program 5 times using all 8 algorithms and 5 more times using the adaptive algorithm (50% of the values were used to select the best compressor). The average time required to compute one 4-step forecast was 9945.46 s. in the former case and 564.63 s. in the latter. Thus, we reduced the computation time by more than 17 times without loss of accuracy.

Table 4. Average absolute errors of the forecasts of the mean monthly sunspot numbers.

Step	1	2	3	4
ppmd	7.2	9.2	10.1	10.2
zstd	8.1	10.3	11.8	13.3
ppmd+zstd	8.1	10.3	11.8	13.3
The joint method	8.1	10.3	11.8	13.3
SWS	8.3	9.1	9.7	9.8

Table 5. Average absolute errors of the forecasts of the mean monthly sunspot numbers (the number of predictions increased by 405).

Step	1	2	3	4
ppmd	16.5	19.1	20.4	21.6
zstd	16.4	18.7	20.0	22.1

4. Model Limitations

The presented model has some limitations. First, it is suitable for prediction of data with no trend pattern. But, as was shown in Section 2.4, to overcome this limitation some additional techniques such as differencing or time series decomposition [28] can be used. Secondly, the computational complexity of the model is high: if we make forecast for h steps ahead, $|A|^h$ sequences have to be compressed. This means that it cannot be directly applied in long-term forecasting or in a setting when it is required to compute predictions very quickly. While sacrificing accuracy, we can still make long-term forecasts using the following approach. Suppose we have a series $X = x_1, x_2, \ldots, x_t$ and we want to predict its next h values. For simplicity, assume that t and h are even numbers, but the generalization is obvious. We can split X into two separate time series $X_{\text{odd}} = x_1, x_3, \ldots, x_{t-1}$ and $X_{\text{even}} = x_2, x_4, \ldots, x_t$. Then we predict $x_{t+1}, x_{t+3}, \ldots, x_{t+h-1}$ using X_{odd} and $x_{t+2}, x_{t+4}, \ldots, x_{t+h}$ using X_{even}. This allows us reduce the number of sequences to compress from $|A|^h$ to $2|A|^{h/2}$. It is clear that we can go further and split X into more than two series. Another obvious way to speed up computations is to choose small n when quantizing.

5. Discussion

In our opinion, the results of computations show that the proposed method has good accuracy. The adaptive approach to forecasting can significantly reduce computation time without loss in effectiveness if multiple compressors are used.

It is important to note that some time series can contain complex regularities. For example, the dynamics of financial time series can be influenced by the participants of the corresponding processes, and this can lead to the emergence of subtle patterns in the data. Another example is the time series arising in the study of space objects. Such series can contain various nonstationarities like attenuation and non-periodic changes. Thus, in our

opinion, forecasting methods that allow finding unusual patterns may be of interest to many researchers. Patterns of the indicated types that occur in text data can be found by modern archivers. Therefore, we believe that the use of data compressors implicitly allows the use of various techniques of finding patterns, including artificial intelligence methods beyond neural networks.

In further research, a more elaborate strategy for optimal algorithms selection in the adaptive method can be developed. For example, one can use multidimensional optimization techniques for this purpose.

6. Conclusions

It turns out that from a mathematical point of view, data compression and prediction can be thought of together, allowing ideas from one area to be used in another. From a practical point of view, many of the data compression techniques implemented in software can be used in time series forecasting. Efficient data compression algorithms to be developed in the future can be combined with existing ones using the described approach.

Author Contributions: Conceptualization, B.R.; methodology, B.R.; software, K.C.; validation, B.R., K.C.; writing—review and editing, K.C., B.R.; Both authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by RFBR, project numbers 19-37-90009, 19-47-540001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://novosibstat.gks.ru/folder/31734 (average number of unemployed persons in Novosibirsk region), https://novosibstat.gks.ru/folder/31847 (average annual monetary income of the population in Novosibirsk region), http://www.sidc.be/silso/datafiles (monthly mean total sunspot numbers), http://listserver.ips.gov.au/pipermail/ips-ssn-predictions/ (SWS observed and predicted solar indices).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hyndman, R.; Koehler, A.B.; Ord, J.K.; Snyder, R.D. *Forecasting with Exponential Smoothing: The State Space Approach*; Springer: Berlin/Heidelberg, Germany, 2008.
2. Shumway, R.H.; Stoffer, D.S. *Time Series Analysis and Its Applications: With R Examples*; Springer International Publishing: Cham, Switzerland, 2017.
3. Tang, Z.; De Almeida, C.; Fishwick, P.A. Time series forecasting using neural networks vs. Box-Jenkins methodology. *Simulation* **1991**, *57*, 303–310. [CrossRef]
4. Zhang, G.P.; Kline, D.M. Quarterly time-series forecasting with neural networks. *IEEE Trans. Neural Netw.* **2007**, *18*, 1800–1814. [CrossRef]
5. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent neural networks for time series forecasting: Current status and future directions. *Int. J. Forecast.* **2020**, *37*, 388–427. [CrossRef]
6. Ryabko, B.Y. Prediction of random sequences and universal coding. *Probl. Inf. Transm.* **1988**, *24*, 87–96.
7. Ryabko, B.; Astola, J.; Malyutov, M. *Compression-Based Methods of Statistical Analysis and Prediction of Time Series*; Springer International Publishing: Cham, Switzerland, 2016.
8. Ryabko, B. Compression-Based Methods for Nonparametric Prediction and Estimation of Some Characteristics of Time Series. *IEEE Trans. Inf. Theory* **2009**, *55*, 4309–4315. [CrossRef]
9. Burrows, M.; Wheeler, D.J. A block-sorting lossless data compression algorithm. In *Tech. Rept. 124*; Digital SRC: Palo Alto, CA, USA, 1994.
10. Cleary, J.; Witten, I. Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun.* **1984**, *32*, 396–402. [CrossRef]
11. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343. [CrossRef]
12. Ziv, J.; Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory* **1978**, *24*, 530–536. [CrossRef]
13. Kieffer, J.C.; Yang, E.H. Grammar-based codes: A new class of universal lossless source codes. *IEEE Trans. Inf. Theory* **2000**, *46*, 737–754. [CrossRef]

14. Nevill-Manning, C.G.; Witten, I.H. Identifying hierarchical structure in sequences: A linear-time algorithm. *J. Artif. Intell. Res.* **1997**, *7*, 67–82. [CrossRef]
15. Sakamoto, H.; Maruyama, S.; Kida, T.; Shimozono, S. A space-saving approximation algorithm for grammar-based compression. *IEICE Trans. Inf. Syst.* **2009**, *92*, 158–165. [CrossRef]
16. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; Wiley-Interscience: Hoboken, NJ, USA, 2006.
17. GNU Project. Gzip. Free Software Foundation. Available online: https://www.gnu.org/software/gzip/ (accessed on 20 June 2020).
18. Tsay, R.S. *Multivariate Time Series Analysis: With R and Financial Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
19. De Silva, A.; Hyndman, R.J.; Snyder, R. The vector innovations structural time series framework: A simple approach to multivariate forecasting. *Stat. Model.* **2010**, *10*, 353–374. [CrossRef]
20. Shih, S.Y.; Sun, F.K.; Lee, H.Y. Temporal pattern attention for multivariate time series forecasting. *Mach. Learn.* **2019**, *108*, 1421–1441. [CrossRef]
21. Wang, K.; Li, K.; Zhou, L.; Hu, Y.; Cheng, Z.; Liu, J.; Chen, C. Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing* **2019**, *360*, 107–119. [CrossRef]
22. Maruyama, S.; Sakamoto, H.; Takeda, M. An online algorithm for lightweight grammar-based compression. *Algorithms* **2012**, *5*, 214–235. [CrossRef]
23. Bille, P.; Gørtz, I.L.; Prezza, N. Space-efficient re-pair compression. In Proceedings of the 2017 Data Compression Conference (DCC), Snowbird, UT, USA, 4–7 April 2017; pp. 171–180.
24. Chirikhin, K.S.; Ryabko, B.Y. Application of artificial intelligence and data compression methods to time series forecasting. In Proceedings of the Applied Methods of Statistical Analysis, Statistical Computation and Simulation-AMSA'2019, Novosibirsk, Russia, 18–20 September 2019; pp. 553–560.
25. Ryabko, B. Time-Universal Data Compression. *Algorithms* **2019**, *12*, 116. [CrossRef]
26. SILSO, World Data Center. Sunspot Number and Long-Term Solar Observations. Royal Observatory of Belgium, On-Line Sunspot Number Catalogue. 1749–2020. Available online: http://www.sidc.be/SILSO/ (accessed on 20 June 2020).
27. Hathaway, D.H. The solar cycle. *Living Rev. Sol. Phys.* **2015**, *12*, 4. [CrossRef] [PubMed]
28. Cleveland, R.B.; Cleveland, W.S.; McRae, J.E.; Terpenning, I.J. STL: A seasonal-trend decomposition procedure based on loess. *J. Off. Stat.* **1990**, *6*, 3–33.