Genetic Algorithm Search for Predictive Patterns in Multidimensional Time Series

Arnold Polanski

School of Management and Economics
Queen's University of Belfast
25 University Square
Belfast BT7 1NN, United Kingdom
a.polanski@qub.ac.uk

Based on an algorithm for pattern matching in character strings, a pattern matching machine is implemented that searches for occurrences of patterns in multidimensional time series. Before the search process takes place, time series data is encoded in user-designed alphabets. The patterns, on the other hand, are formulated as regular expressions that are composed of letters from these alphabets and operators. Furthermore, a genetic algorithm is developed to breed patterns that maximize a user-defined fitness function. In an application to financial data, it is shown that patterns bred to predict high exchange rates volatility in training samples retain statistically significant predictive power in validation samples.

1. Introduction

This work is a contribution to the rapidly developing research area of data mining, a host of methods that aim at revealing hidden relationships and regularities in large sets of data. Of particular importance is the class of data mining problems concerned with discovering frequently occurring patterns in sequential data. We propose a versatile nonparametric technique for representing multidimensional data by encoding it in alphabets that are defined by an analyst user. The encoded data is explored by means of patterns, which are composed of operators and letters from these alphabets. Since patterns are regular expressions, they can be automatically manipulated, combined, and evaluated. These operations lie at the heart of our genetic algorithm (GA), which evolves patterns in order to breed ever better descriptors and predictors of the data. A concise and flexible pattern description language is, therefore, a powerful tool for data mining that serves two purposes: on the one hand, as a language in which theories concerned with the underlying data generating process are formulated and tested and, on the other, as a forecasting instrument.

The present approach shows its special strength when dealing with multidimensional data that can be analyzed under multiple criteria and/or characterized by several indicators. Usually, each criterion

https://doi.org/10.25088/ComplexSystems.19.3.195

(indicator) forms the base of an alphabet. Preprocessing the data by encoding it in alphabets ensures that the search for patterns unfolds efficiently. This is manifestly a precondition for a viable GA application when the algorithm evaluates patterns based on their matches. Furthermore, the possibility to design data-specific alphabets makes the method applicable not only to highly diverse record sets but also allows each researcher to analyze the (same) data with an idiosyncratic language. We stress here an important departure from the more traditional techniques of forecasting complex systems. Many methods, like kernel regression, neural networks, or reinforcement learning (see [1] for recent developments), estimate the future output of a system as a function of a fixed number of past observations. In contrast, the present approach does not restrict the "relevant past" to time windows of fixed lengths. It is only important that the past state of the system and the system's response to that state frequently generate measurable outcomes with some, typically ex ante unknown, characteristics. These characteristics are encapsulated as patterns in a suitable language and searched for in the encoded time series.

The analysis and forecasting of multidimensional data is at the center of research in, for example, finance, electrical engineering, theoretical physics, and the computer sciences. Various mathematical methods have been proposed for the description and analysis of interdependencies in multivariate time series. An interested reader can find a recent overview of these methods (including Granger causality, directed transfer functions, and partial directed coherence) in [2]. In this work we are interested in an ex ante unknown type of multidimensional relationship with possibly changing time frames; as such, a less structured methodology is required. As a well established and versatile search heuristic, GAs seem to be a promising approach for generating pattern descriptors with predictive power. Although mathematical foundations and properties of GAs are far from being settled, there is some evidence that GAs might become a generic tool for universal computation. For example, the work by Sapin et al. [3, 4] suggests that GAs have the potential to identify cellular automata that support universal computation.

This paper is organized as follows: in Section 2, we describe the encoding process for multidimensional time series and define patterns. The GA for pattern evolution is presented in Section 3. In Section 4, some related approaches are discussed. Section 5 contains an application to financial time series data and Section 6 concludes.

2. Time Series, Texts, and Patterns

Based on an algorithm for pattern matching in character strings [5], we implement a deterministic pattern matching machine that searches for occurrences of patterns in multidimensional time series

https://doi.org/10.25088/ComplexSystems.19.3.195

$x = (x^1, \ldots, x^N)$, where $x^i = (x_1^i, \ldots, x_T^i)'$ is a vector of T observations. Before the search process takes place, the time series data is encoded as strings of letters from user-defined alphabets. Alphabets are sets composed of mathematical expressions (conditions) that yield the Boolean value true or false when evaluated with respect to x. For example, the condition $x_t^i > x_{t-1}^i$ returns true (false) at all dates t at which the i^{th} time series increases (weakly decreases). A sequence of conditions is called an *alphabet* if exactly one of them is true in each period t. Hence,

$$\left\{x_t^3 > x_{t-1}^3, x_t^3 \leq x_{t-1}^3\right\}$$

is an example of an alphabet with two mutually exclusive conditions-letters. Given a multidimensional time series x and a set of alphabets $\{A^1, \ldots, A^K\}$, the following algorithm generates a $T \times K$ dimensional text $a = (a_t^k)$:

for each date $t = 1, \ldots, T$
  for each alphabet $k = 1, \ldots, K$
    a_t^k = the ordinal of a condition in A^k
      that evaluates true with regard to x at t.

Each column a^k in the text a represents the information in x that is encoded through A^k. A generic element a_t^k is an integer between one and the number of conditions-letters in the alphabet A^k. We consider, therefore, the matrix a as a multidimensional text recorded in natural numbers. Note that the number of time series N in x and the number of alphabets K will usually differ. The diagram in Figure 1 illustrates the encoding of a fragment of five observations from a multidimensional time series $x = (x_t^1, \ldots, x_t^4)$ according to two alphabets $\{A^1, A^2\}$ with the resulting bidimensional text $a = (a_t^1, a_t^2)$ for $t = 1, \ldots, 5$.

The choice of alphabets is entrusted to the expertise of the end user of the system. Generally, the letters in the alphabets will test conditions on certain indicators that are deemed relevant for the subject under study. The latter indicators may be derived from economic variables in early warning systems for the prediction of financial crises [6, 7], from specific protein information in cancer detection systems [8, 9], or from technical trading rules [10–13]. The almost unrestricted freedom in the specification of alphabets is at the same time a strength and a weakness of the present approach. On the one hand, its inherent flexibility allows for immediate and fine-tuned application to many research areas but, on the other, it burdens the researcher with a tedious and ultimately open question of finding optimal alphabets. In Section 5, we illustrate the types of alphabets that can be used for encoding financial data.

https://doi.org/10.25088/ComplexSystems.19.3.195

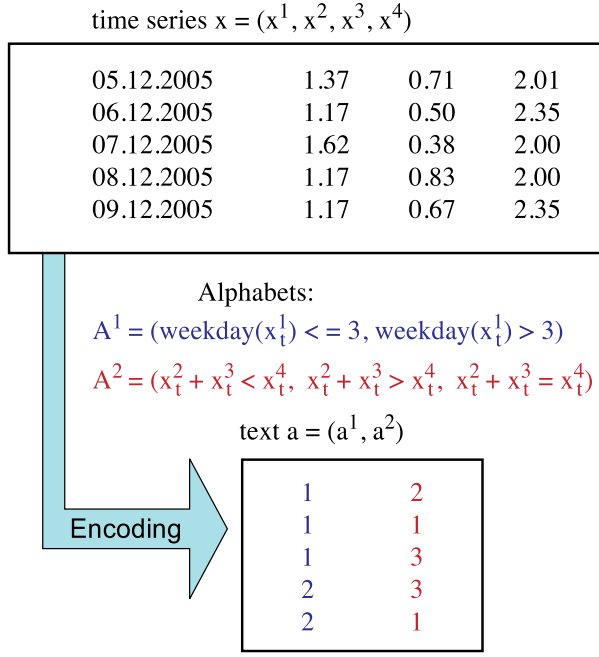

Figure 1. The encoding of a time series that includes dates. The time series x^1 contains the dates 5.12.2005 through 9.12.2005 that correspond to weekdays 1 through 5 (Monday through Friday).

Given a set of alphabets, a succinct description of a relevant aspect of the underlying data is expressed as a pattern. The latter is defined as a regular expression that is composed of letters, operators, and parentheses. A letter is represented by a pair [condition-letter number : alphabet] enclosed in square brackets. For example, [2 : 1] stands for the second condition-letter from the first alphabet. We consider the three fundamental operators concatenation, or, and and.

- *Concatenation*, as used, for instance, in the pattern [2 : 1] [3 : 2] where the second letter from the first alphabet is followed by the third letter from the second alphabet. Since the text a consists of natural numbers and $a_t^k = i$ is interpreted as the i^{th} letter of the k^{th} alphabet at position t, this pattern matches fragments of a, starting at t, such that $a_t^1 = 2$ and $a_{t+1}^2 = 3$.
- *Or* (+) between two subpatterns P_1 and P_2 implies that there is a match if and only if either P_1 or P_2 (or both) occurs. For example, the pattern [3 : 1] [5 : 1] + [4 : 2] [4 : 2] describes fragments of the text a, where $a_t^1 = 3$ is followed by $a_{t+1}^1 = 5$ or where $a_t^2 = a_{t+1}^2 = 4$.

https://doi.org/10.25088/ComplexSystems.19.3.195

- *And* $(*)$ between two subpatterns P_1 and P_2 implies that there is a match if and only if both P_1 and P_2 occur simultaneously. The pattern $[3:1]*[4:2][6:2]$ detects, therefore, fragments of a such that $a_t^1 = 3$ and $a_t^2 = 4$, $a_{t+1}^2 = 6$.

Finally, parentheses induce the desired order of operators in the usual way:

$$([3:1]+[4:1])*[5:2] \equiv \left(a_t^1 = 3 \text{ or } a_t^1 = 4\right) \text{and } a_t^2 = 5,$$
$$[3:1]+([4:1]*[5:2]) \equiv a_t^1 = 3 \text{ or } \left(a_t^1 = 4 \text{ and } a_t^2 = 5\right).$$

A pattern that complies with the syntactic rules can be searched for in the encoded text. The following algorithm searches for matches of the pattern p in the text a between dates T_1 and T_2 (i.e., in all rows of a between and including a_{T_1} and a_{T_2}):

set $t = T_1$;
while $t \le T_2$ repeat
{ if a match of length k starts at date t then
record $(t, t+k-1)$ in the set $M_p(T_1, T_2)$;
set $t = t+1$; }.

The outcome of the algorithm is exemplified in Figure 2, where the fragments of the text a that are matched by the pattern specification $p = [1:1]*([2:2]+[4:2][5:3])$ are enclosed in rectangles (the first rectangle matches $[1:1]*[2:2]$ while the second matches $[1:1]*([4:2][5:3])$).

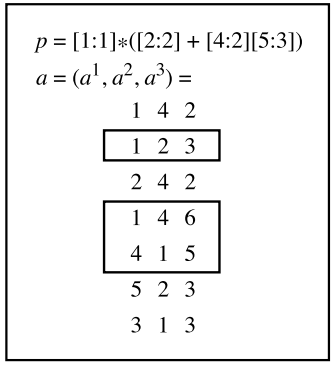

Figure 2. Fragments of the text a that are matched by the pattern p.

The matching algorithm itself is based on the implementation of the deterministic finite state automaton in [5] with important modifications to account for multidimensional texts and operators. After

https://doi.org/10.25088/ComplexSystems.19.3.195

running the matching program, the set $M_p(T_1, T_2)$ contains pairs (t_s, t_e) with the start and end dates of matches. If two or more matches start on the same date, the match with the minimum length is recorded. If two or more matches end at the same time, only one of them is kept in $M_p(T_1, T_2)$.

The elements of $M_p(T_1, T_2)$ will be typically used as signals of something, say, a financial crisis, a share price increase, or a tissue developing cancer. In order to evaluate the predictive power of patterns, the user must define a fitness function that maps the set of matches into real numbers. For example, if the vector x^1 contains stock prices at consecutive decision times (i.e., days or hours), then the function

$$\sum_{(t_s,t_e)\in M_p(T_1,T_2)} \ln\left(\frac{x^1_{t_e+1}}{x^1_{t_e}}\right)$$

computes the accumulated profit that is made when the stock is bought, whenever a match ends in the decision period. This function attains high values for patterns that consistently signal rising stock prices after match occurrences. It can be interpreted, therefore, as a measure of fitness for patterns that act as buy signals. An interesting point to note is that the patterns are searched for in the encoded text a, while the evaluation of the pattern fitness involves the original time series x.

Besides the use as a forecasting instrument, the present approach may be applied as a language, in which quantitative theories are formulated. Suppose, for instance, that a researcher conjectures that a variable x^1 under study exceeds a desired level $\overline{x}^1$ if either the variable x^2 assumes values below $\overline{x}^2$ or the variable x^3 assumes values above $\overline{x}^3$. Then, after encoding $x = \left(x^1, x^2, x^3\right)$ in three alphabets,

$$A^k = \left\{x^k_t \le \overline{x}^k, x^k_t > \overline{x}^k\right\},\ k = 1,\ ..,\ 3,$$

for some thresholds $\overline{x}^k$, the latter theory can be phrased in terms of the pattern

$$[2:1] * ([1:2] + [2:3])$$

and matched with the data.

In Section 3 we develop a GA for pattern evolution. The aim of the GA is to create a population of patterns that are optimized with respect to a fitness function over a training set. Obviously, patterns bred in training samples will be reliable predictors only if they retain their predictive power in evaluation samples.

https://doi.org/10.25088/ComplexSystems.19.3.195

3. Genetic Algorithm

A GA is a search technique for finding approximate solutions to optimization and search problems. GAs are typically implemented as a computer simulation in which a population of abstract representations (chromosomes) of candidate solutions to an optimization problem evolves toward better solutions. The present GA evolves patterns, that is, regular expressions that use the building blocks of letters, parentheses, and operators. By combining and modifying the best performing parent patterns, new generations of offspring with increasing average fitness are created. The present GA uses the three basic operations of crossover, mutation, and selection.

- *Crossover* (xover) extracts fragments from two parent patterns and combines them by means of fundamental operators into a valid offspring pattern as illustrated in the following example:

  $$([1:1]+[1:2])[2:3]\ \text{xover}\ [2:1]\,[2:2]*[3:1]$$
  $$\rightarrow\ ([1:1]+[1:2])+[3:1].$$

  In this example, a combination of $([1:1]+[1:2])$ and $[3:1]$ is inherited by the offspring, while $[2:3]$ and $[2:1]\,[2:2]$ vanish.

- *Mutation* changes a part of the pattern to a subpattern that matches a fragment randomly drawn from the encoded text a. In the next example, the expression in parentheses undergoes a mutation to the subpattern $[1:2]*[3:3]$ that matches $a_t^2 = 1$, $a_t^3 = 3$, a randomly retrieved fragment of a:

  $$[1:1]\,[2:1]\,([1:1]+[3:2])\ \rightarrow_{\text{mut}}\ [1:1]\,[2:1]\,([1:2]*[3:3]).$$

- *Selection* picks out the best-performing patterns (with respect to the user-defined fitness function).

Note that the result of the breeding process is a regular expression that complies with the syntactic rules for patterns. The structure of the main loop of the GA is depicted in Figure 3.

https://doi.org/10.25088/ComplexSystems.19.3.195

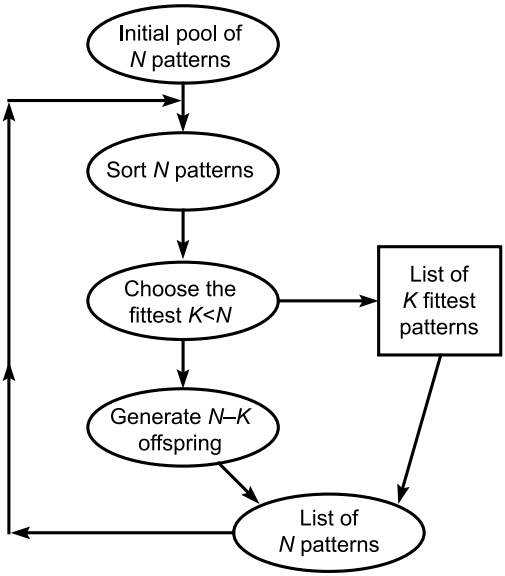

Figure 3. The structure of the main loop of the pattern breeding GA.

4. Related Literature

Faced with abundant literature on data mining, we focus on three closely related papers in order to emphasize the main departures of the present work from the existing approaches.

Szpiro [13] implements a GA that permits the discovery of equations of the data-generating process in symbolic form. His GA uses parts of equations, constants, and the basic arithmetic operators to breed ever better formulas. Apart from furnishing a deeper understanding of the dynamics of a process, his method also permits global predictions and forecasts. Unlike his search for a hidden relationship, our GA does not work on raw data but on encoded information. This approach allows for including predictors (e.g., adaptive moving averages) that are very unlikely or impossible to be developed by Szpiro's algorithm. Furthermore, his algorithm is restricted to uncovering functional relationships whereas ours detects relevant patterns in data.

Dempster [14] applies a GA to evolve trading rules that are based on technical indicators. Potential rules are constructed as binary trees in which the terminal nodes are indicators (e.g., adaptive moving averages, relative strength index, stochastics, or momentum oscillators) yielding a Boolean signal at each time step, and the nonterminal nodes are the Boolean operators AND, OR, and XOR. The result

https://doi.org/10.25088/ComplexSystems.19.3.195

of this procedure is a set of fittest trading rules that recommend a transaction (buy or sell) in each period. Unlike trading rules, patterns are not constrained to emit a buy/sell signal at each time step. They are more flexible in the sense that they can focus exclusively on informative sequences of observations. Furthermore, the algorithm in [14] (and other commonly used algorithms for information extraction) work with data windows of fixed length. The GA described in Section 3 breeds patterns without knowing the number of observations that they match at the time of the design. Hence, it is able to create patterns that are able to detect regularities which emerge after specific histories. In this manner, qualitatively identical phenomena that unfold on different time scales (fractal patterns) or stretch over time windows of variable length can be captured.

Finally, Packard [15] develops a GA that evolves a population of conditions, defined on an unidimensional independent variable x, as in the following example:

$$C = (20.1 \leq x_t) \wedge (30 \leq x_{t+1} \leq 40.5) \wedge (x_{t+2} \leq 30).$$

Packard's algorithm works on conditions, adjusting constants and operators in order to obtain good predictors for a dependent variable. This approach is similar in spirit to evolving expressions, composed of conditions-letters, as described in Section 2. Nevertheless, an attempt to include elaborated indicators into Packard's conditions leads to intolerable runtimes as they must be evaluated during the matching phase for each date t. Furthermore, an obvious extension of Packard's GA to multidimensional time series suffers severely from the curse of dimensionality.

5. An Application

5.1 Data and Alphabets

As an application, we tested the predictive power of patterns on financial time series data. We used the daily exchange rates for several currency pairs. The data was downloaded from
http://www.forexrate.co.uk/forexhistoricaldata.php.

For each pair and day t, the vector $x_t = \left(x_t^i\right)^{i=1,\ldots,5}$ contained

$$x_t^1 = \text{date},\ x_t^2 = \text{open},\ x_t^3 = \text{close},\ x_t^4 = \text{min},\ x_t^5 = \text{max},$$

that is, the current date and the opening, closing, maximum, and minimum exchange rates during this day. We used 1201 weekday observations from August 25, 2003 through April 12, 2008 (hence, $x = \left(x_t^i\right)$ had the dimensions 1201×5). We encoded x according to six alphabets obtaining a text a that is six-dimensional and 1200 characters long:

https://doi.org/10.25088/ComplexSystems.19.3.195

$$A^1 = \{x_t^1 = \text{Monday}, \ldots, x_t^1 = \text{Friday}\},$$

$$A^2 = \left\{\frac{x_t^3}{x_{t-1}^3} < 0.998,\ 0.998 \le \frac{x_t^3}{x_{t-1}^3} < 0.9985,\ \ldots,\ \frac{x_t^3}{x_{t-1}^3} \ge 1.002\right\},$$

$$A^3 = \left\{\frac{x_t^3}{x_t^4} < 1.0005,\ 1.0005 \le \frac{x_t^3}{x_t^4} < 1.001,\ \ldots,\ \frac{x_t^3}{x_t^4} \ge 1.005\right\},$$

$$A^4 = \left\{\frac{x_t^5}{x_t^3} < 1.0005,\ 1.0005 \le \frac{x_t^5}{x_t^3} < 1.001,\ \ldots,\ \frac{x_t^5}{x_t^3} \ge 1.005\right\},$$

$$A^5 = \left\{\frac{x_t^3}{x_t^2} < 0.996,\ 0.996 < \frac{x_t^3}{x_t^2} \le 0.997\right],\ \ldots,\ \frac{x_t^3}{x_t^2} \ge 1.004\right\},$$

$$A^6 = \left\{\frac{x_t^5}{x_t^4} < 1.0005,\ 1.0005 \le \frac{x_t^5}{x_t^4} < 1.001,\ \ldots,\ \frac{x_t^5}{x_t^4} \ge 1.005\right\}.$$

All alphabets except A^1 were composed of nine conditions-letters and all of them used only past and present information in x. In particular, each row a_t of the text was generated by accessing information in x_{t-1} and x_t only. The requirement of using only available information is, obviously, essential when we test the predictive power of patterns.

5.2 Pattern Evaluation and the Fitness Function

In order to create effective patterns by means of a GA in-sample, and to assess their predictive power out-of-sample, a suitable definition of the fitness function is crucial. The fitness function that we employed was designed to measure the difference between sample means for two mutually exclusive and collectively exhaustive sets: the set of end-dates of matches and its complement. Specifically, for each pattern p that was matched in the time window $[T_1, T_2]$, we partitioned this window into two groups: the subset M of end-dates of p-matches and its complement NM. In each subset, we computed the sample mean and the sample variance for the next day log-returns,

$$\overline{x}_m = \sum\nolimits_{t \in M} \frac{\ln\left(\frac{x_{t+1}^3}{x_{t+1}^2}\right)}{n_m},$$

$$\overline{x}_{nm} = \sum\nolimits_{t \in \mathrm{NM}} \frac{\ln\left(\frac{x_{t+1}^3}{x_{t+1}^2}\right)}{n_{nm}},$$

https://doi.org/10.25088/ComplexSystems.19.3.195

$$s_m^2 = \sum_{t \in M} \frac{\left(\ln\left(\frac{x_{t+1}^3}{x_{t+1}^2} \right) - \overline{x}_m \right)^2}{n_m - 1},$$

$$s_{nm}^2 = \sum_{t \in \mathrm{NM}} \frac{\left(\ln\left(\frac{x_{t+1}^3}{x_{t+1}^2} \right) - \overline{x}_{\mathrm{nm}} \right)^2}{n_{nm} - 1},$$

where $n_m = |M|$ and $n_{nm} = |\mathrm{NM}|$.

For these values, we calculated the difference-of-means statistic,

$$t_{\mathrm{return}} = \frac{\overline{x}_m - \overline{x}_{nm}}{\sqrt{s_m^2 / n_m + s_{nm}^2 / n_{nm}}},$$

and used it as both the fitness function for pattern breeding in the training set and as an estimate of predictive power in the validation set. The fitness function favored patterns indicating relatively high expected returns for the next day. Should the evolved patterns retain high fitness out-of-sample, our approach would be a (statistically) effective forecasting instrument.

We applied the same procedure to define the performance measure t_{range} to evaluate patterns with respect to the next-day log difference in intraday extreme (min and max) values $\ln\left(x_{t+1}^5 / x_{t+1}^4\right)$. Parkinson [16] proposed the difference in extreme values as a proxy for volatility. We therefore considered patterns evolved with t_{range} as indicators of high volatility.

5.3 Genetic Algorithm

After encoding the data and defining the fitness function, we run a number of GA experiments. The main loop of each GA experiment (see Figure 3) evolved a population of $N = 100$ patterns, out of which the elite of $K = 15$ fittest survived each round and were selected to reproduce. Each breeding loop was repeated 50 000 times using a training window of 800 observations to compute the fitness. Subsequently, the single best performing pattern of the breeding stage was tested in an out-of-sample (validation) window of 400 observations. We experimented with different parameter values for the GA operators without, however, detecting a significant impact on the results. Furthermore, in order to avoid overfitting in the training set, we allowed only patterns with at least 10 matches per 100 observations to survive.

https://doi.org/10.25088/ComplexSystems.19.3.195

5.4 Results

Table 1 summarizes the results of the GA experiment, which were computed as averages over 10 runs. Broadly speaking, Table 1 confirms the well-known stylized facts that the returns are not predictable but the volatility is (see [17] for a survey). Specifically, given the high numbers of matches in validation sets (Table 1), we could rely on the central limit theorem and assume that the statistic t_{return} is standard normal under the Null of equal means in the subsets M and NM of the validation set. As the third column in Table 1 shows, the Null could not be rejected at any reasonable significance level for any currency pair. In other words, the best performing pattern in the training set (with the fitness reported in the second column) failed to detect matches in the validation set that were followed by significantly higher log returns for the next day.

On the other hand, the Null was rejected at least at the 1% and, usually, much lower, significance level when tested for the log difference in extreme values with the t_{range} statistic. Only for the pair GBP / USD in the validation set did we obtain the P-value (2.34) $\approx 1\%$. This is probably due to the relatively small number of matches (45) in this set. The next largest P-value in the validation set is of order 10^{-6} ($t_{\text{range}} = 4.36$ for GBP / CHF).

The winning pattern in the training set detected effectively next-day high volatility also out-of-sample (the last column in Table 1). Our approach was, therefore, successful at creating (statistically) reliable predictors of volatility.

Currencies	t_{return} | training	t_{return} | validation	t_{range} | training	t_{range} | validation
GBP / EUR	5.23 (220)	−1.09 (107)	7.14 (202)	5.04 (97)
GBP / USD	5.34 (212)	0.31 (132)	7.20 (172)	2.34 (45)
GBP / CHF	5.90 (163)	−0.45 (71)	7.02 (190)	4.36 (92)
USD / EUR	5.66 (188)	0.77 (163)	7.35 (200)	4.72 (89)

Table 1. In-sample (training) and out-of-sample (validation) t-statistics for one-day-ahead prediction of returns $\ln\left(x_{t+1}^3 / x_{t+1}^2\right)$ (t_{return}) and volatility $\ln\left(x_{t+1}^5 / x_{t+1}^4\right)$ (t_{range}). In parentheses, the number of matches.

To compare our procedure with a standard technique of volatility forecasting, we tested the forecasts generated by the exponentially weighted moving average (EWMA). EWMA is widely used in practice due to its simplicity and its reported superiority over more sophisticated models [18]. The EWMA specifies the next period's volatility v_{t+1} as a weighted average of the current modeled volatility v_t and the current observed volatility, here measured by the price range

https://doi.org/10.25088/ComplexSystems.19.3.195

$\ln\left(x_t^5 \,/\, x_t^4\right)$:

$$v_{t+1} = \alpha\, v_t + (1-\alpha)\ln\left(\frac{x_t^5}{x_t^4}\right).$$

For the same data as in the GA experiment, we estimated the EWMA parameter α in the training window of 800 observations by the maximum likelihood method and specified the threshold t_v of "high volatility". This threshold was set equal to the third quartile of the empirical distribution of observed volatilities in order to create a similar number of high volatility days as in the GA experiment, that is, roughly 1/4 of all observations in the sample. Using the given specifications, we partitioned the validation set of 400 observations into two groups: the subset M of high volatility forecasts, $v_{t+1} > t_v$, and its complement NM. For each subset, we computed the sample mean and the sample variance of observed volatilities $\ln\left(x_{t+1}^5 \,/\, x_{t+1}^4\right)$ and calculated the difference-of-means statistic. The results, reported in Table 2, indicate that EWMA forecasts of high volatility are statistically significant, although (with the exception of the GBP / USD pair) the t-statistics lie below the values from the GA experiment (Table 1). In this simple example, the parsimony of the EWMA approach may outweigh its slightly worse performance as compared to the elaborate GA procedure. The latter procedure, however, is designed to detect complicated multidimensional relationships where its full strength can come to the fore.

Currencies	t_{range} | training	t_{range} | validation
GBP / EUR	6.27 (202)	4.81 (97)
GBP / USD	5.80 (172)	3.73 (45)
GBP / CHF	5.22 (190)	3.76 (92)
USD / EUR	6.05 (200)	4.28 (89)

Table 2. In-sample (training) and out-of-sample (validation) t-statistics for one-day-ahead EWMA forecasts. The forecasts were computed with the ML estimate $\hat{\alpha} = 0.94$. In parentheses, the number of "high volatility" days.

6. Conclusions

Based on an algorithm for pattern matching in character strings, we implement a pattern-matching machine that searches for occurrences of specified patterns in multidimensional time series. Before the search process takes place, the time series are encoded as strings of letters from user-defined alphabets. The preprocessing of the raw data has conceptual advantages and also speeds up the matching phase deci-

https://doi.org/10.25088/ComplexSystems.19.3.195

sively. Since the evaluation of patterns is based on their matches, an efficient matching algorithm is essential for creating optimal patterns by means of a genetic algorithm (GA). The GA combines parent patterns in order to breed offspring (randomly modified by mutations) that are ever better predictors. In an application to financial time series, we show that the presented GA has the potential to produce patterns with significant out-of-sample predictive power.

References

[1] W. Wobcke and M. Zhang, eds., *Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence (AI 2008)*, Auckland, New Zealand, Berlin: Springer-Verlag, 2009.

[2] R. Dahlhaus, J. Kurths, P. Maass, and J. Timmer, eds., *Mathematical Methods in Time Series Analysis and Digital Image Processing*, Springer, 2008.

[3] E. Sapin, O. Bailleux, and J. Chabrier, "Research of Complexity in Cellular Automata through Evolutionary Algorithms," *Complex Systems*, **17**(3), 2007 pp. 231–241

[4] E. Sapin and L. Bull, "Evolutionary Search for Cellular Automata Logic Gates with Collision-Based Computing," *Complex Systems*, **17**(4), 2008 pp. 321–338.

[5] R. Sedgewick, *Algorithms*, Reading, MA: Addison-Wesley, 1988.

[6] F. X. Diebold and G. D. Rudebusch, "Scoring the Leading Indicators," *Journal of Business*, **62**(3), 1989 pp. 369–91.

[7] G. Kaminsky, S. Lizondo, and C. Reinhart, "Leading Indicators of Currency Crises," *IMF Staff Papers*, **45**(1), 1998 pp. 1–48.

[8] B. L. Adam et al., "Serum Protein Fingerprinting Coupled with a Pattern-Matching Algorithm Distinguishes Prostate Cancer from Benign Prostate Hyperplasia and Healthy Men," *Cancer Research*, **62**(13), 2002 pp. 3609–3614.

[9] E. F. Petricoin et al., "Serum Proteomic Patterns for Detection of Prostate Cancer," *Journal of the National Cancer Institute*, **94**(20), 2002 pp. 1576–1578.

[10] H. Iba and N. Nikolaev, "Genetic Programming Polynomial Models of Financial Data Series," in *Proceedings of the 2000 Congress on Evolutionary Computation (CEC '00)*, La Jolla, CA, New York: IEEE Press, 2000 pp. 1459–1466.

[11] R. Levich and L. Thomas, "The Significance of Technical Trading-Rule Profits in the Foreign Exchange Market: A Bootstrap Approach," *Journal of International Money and Finance*, **12**(5), 1993 pp. 451–474.

[12] C. Neely, P. Weller, and R. Dittmar, "Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach," *Journal of Financial and Quantitative Analysis*, **32**(4), 1997 pp. 405–426.

https://doi.org/10.25088/ComplexSystems.19.3.195

[13] G. Szpiro, "A Search for Hidden Relationships: Data Mining with Genetic Algorithms," *Computational Economics*, **10**(3), 1997 pp. 267–277.

[14] M. A. H. Dempster, T. W. Payne, Y. Romahi, and G. W. P. Thompson, "Computational Learning Techniques for Intraday FX Trading Using Popular Technical Indicators," in *IEEE Transactions on Neural Networks*, **12**(4), 2001 pp. 744–754.

[15] N. Packard, "A Genetic Learning Algorithm for the Analysis of Complex Data," *Complex Systems*, **4**(5), 1990 pp. 543–572.

[16] M. Parkinson, "The Extreme Value Method for Estimating the Variance of the Rate of Return," *Journal of Business*, **53**(1), 1980 pp. 61–65.

[17] T. Bollerslev, R. Chou, and K. Kroner, "ARCH Modeling in Finance: A Review of the Theory and Empirical Evidence," *Journal of Econometrics*, **52**(1–2), 1992 pp. 5–59.

[18] C. Guermat and R. D. F. Harris, "Forecasting Value at Risk Allowing for Time Variation in the Variance and Kurtosis of Portfolio Returns," *International Journal of Forecasting*, **18**(3), 2002 pp. 409–419.

https://doi.org/10.25088/ComplexSystems.19.3.195