Pattern Matching in Multidimensional Time Series

Arnold Polanski[*]
Facultad de Económicas, Universidad de Alicante, E-03071 Alicante, Spain
arnold@merlin.fae.ua.es

Abstract

Based on the algorithm for pattern matching in a character string, a pattern description language (PDL) is developed. The compilation of a regular expression, that conforms to the PDL, creates a non-deterministic pattern matching machine (PMM) that can be used as a searching device for detecting sequential patterns or functional (statistical) relationships in multidimensional data. As an example, a chart pattern of ex ante unknown length is encoded and its occurences are searched for in financial data.

1. Introduction

"The explanation and the prediction of natural phenomena, be it to forecast the future or to elicit the hidden laws that underlie unknown dynamics, are the ultimate aims of science" ([8], p. 267). A relatively new research area concerned with revealing hidden relationships and regularities in observed variables is data mining, a host of methods that aim at extracting previously unknown and potentially useful information from large sets of data. Of particular importance is the class of data mining problems concerned with discovery of frequently occurring patterns in sequential data (time series). Many systems (e.g. financial markets) that generate sequential data can be seen as feed-back driven, i.e. their past output is one of their important inputs. Standard methods of complex system forecasting, like kernel regression or neural networks, take it into account when trying to estimate the current and future output values as "functions" of a (fixed) number of past data. On the other hand, the present approach does not restrict neither the "relevant past" nor the reaction of the system to a time window of a fixed length. It is only important that the past state of the system and the system´s reaction to that state frequently generate sequences of data with some (ex ante unknown) characteristics. Those characteristics can be encoded as patterns in a suitable language and searched for in a time series. A concise and flexible pattern description language could be a powerful tool for data mining and would serve two purposes: on the one hand as a language in which theories concerned with the underlying data generating process can be formulated and tested, and, on the other, as a forecasting instrument for practical applications.

There is some literature on mining sequential patterns in a database of customer transactions (e.g. [1],[3]). The approach pursued there can be briefly stated as follows: Given a set of data sequences, the problem is to discover subsequences that are frequent, in the sense that the percentage of data sequences containing them exceeds a user-specified minimum support. Perhaps the most closely related work to the present one is Packard [5]. Packard develops a GA to address the problem of predicting complex systems. In the simplest form his approach can be stated as follows (cp. [4]):

- There is a series of observations, $\{(x_1; y_1) \ldots (x_T; y_T)\}$, where x_t is the independent and y_t the dependent variable, $1 \le t \le T$. For example in a stock market prediction task, the independent variables, $x_1 \ldots x_T$, might be the prices of a particular stock at successive days and the dependent variables, $y_1 \ldots y_T$, might be representing the prices of the same stock at some time in the future, $y_t = x_{t+k}$.

- There is a population of conditions on the independent variables that are expected to give good predictions for the dependent variables. Consider for example the following condition,

  $$C = [(20 \le x_t) \wedge (30 \le x_{t+1} \le 40) \wedge (x_{t+2} \le 30)]$$

  which represents all the sets of three successive days in which C was met.

Packard uses GA to search for conditions that are good predictors of something. In particular, he looks for conditions that specify sequences of data points whose dependent-variable values are close to being uniform. The fitness of a condition C is calculated by running all the data points $(x; y)$ in the training set through C and collecting corresponding values of y for each sequence in x that satisfies this condition.

[*] I am grateful to the participants of the workshop organized by F. Vega-Redondo, P. Guinea and E. Louis, held in Alicante in autumn 1999, for showing me new horizons of Economics. I profited inmensly from seminars and discussions, I participated in, at the 6th International Conference on Computing in Economics and Finance in Barcelona 2000. Particulary helpful were the talks with M.A. Kaboudan. Special thanks to J. Sikora who shared with me his rich experience as (technical) trader.

If these values are all close to a particular value $\bar{y}$, then C is a candidate for a good predictor. We refer the interested reader to [5] and [4] for details of the GA used for selection of the fittest conditions and for discussion of the results obtained[1]. In this paper we will extend Packard´s ideas and develop a language for succinct encoding of quite general, nondeterministic conditions or patterns that describe (fragments of) a multidimensional time series.

2. String search in a time series

The problem of finding sequences of data points of a time series satisfying certain class of conditions can be recast as a problem of pattern matching in a character string, commonly used in text editors. To start with, suppose there is a (unidimensional) time series $x = (x_1 \ldots x_T)$, that will play the role of a string, where patterns are searched for. We must now represent the data points in x as a sequence of letters of an alphabet. One obvious possibility is to normalize the values in x, such that all observations lie, for instance, in the interval $[0,1]$. This interval is then divided in subintervals (of constant or variable length) and to each subinterval a "letter" is assigned.

Example 1. In a normalized time series x ten subintervals, [0.0,0.1), [0.1,0.2),..,[0.9,1.0], are created and the letters A..J are assigned to them. A condition $C = [(0.2 \leq x_t < 0.3) \wedge (0.3 \leq x_{t+1} < 0.4)]$ can now be expressed as a pattern P = "CD"; with the meaning, that a match exists, whenever D follows C in the string x (written in the alphabet composed of the letters A..J). Searching for the condition C in time series x is equivalent to looking for the corresponding pattern P in the string x. This is obviously an instance of a simple pattern matching problem.

It is often desirable to do searching with somewhat more general description of the pattern. We will consider, in what follows, pattern descriptions made up of symbols (letters) tied together with the following three fundamental operators:

-concatenation as used for instance in the pattern P="CD". If two characters are adjacent in the pattern, then there is a match iff the same two characters are adjacent in the string.

-or (+) between two letters. There is a match iff either of the letters occurs in the string searched.

-Kleene closure operator (*). If we have the closure of a symbol, then there is a match iff the symbol occurs any number of times in the string.

Example 2. With these operators we are able to do searching for more general, nondeterministic patterns. $P = (A^*) + (J^*)$ looks e.g. for sequences (of any length) of letters A or sequences (of any length) of symbols J (sequences of extremely low or extremely high values in the time series x, written in the alphabet defined in example 1).

3. The pattern description language (PDL)

In this section a definition of a language will be given that allows for a quite general specification of patterns in multivariate time series[2].

Be x a $N \times T$ matrix composed of N vectors, $x_1 .. x_N$, with T data points each vector. x will be the space where patterns will be searched for. The notation $x_{it}; 1 \leq t \leq T$; means the t^{th} observation of the time series $x_i; 1 \leq i \leq N$. For notational convenience we will sometimes write x_t instead of x_{1t}. A pattern description will consist of sequences of conditions, or- and closure-operators and parentheses combined in arbitrarily complicated way. A condition is defined as any expression that involves arithmetical and logical operators, constants and elements of vectors $x_1 .. x_N$ indexed by functions of time index t (user defined functions and variables can also be used). The evaluation of an expression as function of t, and possibly of any other variable, yields the boolean value true or false. The conditions are verified in the order, they are found in the pattern description. After evaluating a condition and before passing to the next one, the time index is increased by 1. This means, in particular, that two conditions adjacent in the pattern description and both containing the term x_t, will refer to adjacent observations in x. To delimit conditions in a pattern description, they will be enclosed in square brackets. A pattern encoded using these rules will be called a regular expression.

Example 3. One of the most useful conditions is one which encodes the rise or fall of a time series. It is simply stated as $[x_t < x_{t+1}]$ or $[x_t > x_{t+1}]$, respectively. If we look for sequence of indefinite length with rising values (that ends when the values stop to increase), we can use a regular expression of the form, $[x_t < x_{t+1}]^*[NOT(x_t < x_{t+1})]$.

To find all pairs of adjacent data points, where the plot of prices of stock A crosses from below the plot of an indicator (for instance, the moving average of prices of A), we write: $[x_{1t} < x_{2t}][x_{1t} > x_{2t}]$, where x_1 is the vector of prices of stock A and x_2 the vector of indicator values.

In practical applications some enhancements of the pattern description language proved useful. First, to measure the number of occurrences of a condition,

[1] In 1991 Packard left the University of Illinois to help form a company to predict financial markets.

[2] At this stage we have to abandon the introductory analogy to pattern definition and pattern searching in character strings.

followed by the closure operator "*", variables l_1::l_n were defined. During the search for the pattern P in x, l_i contains the number of evaluations of the condition followed by the i^{th} asterisk that yielded true, i.e. l_i measures the length of the fragment of data in which this condition was met. l_i¶s are set to zero every time a complete pattern match is found or the search is started from the beginning of the pattern (after possible match failure at some position).

Another extension makes possible the suppression of the time index increase after evaluating an expression at position indicated by that index. To this end the expression has to be enclosed in braces.

Finally, to keep track of the length of a match, the variable t_0, which points to the beginning of that match, is defined.

Example 4. $[x_t < x_{t+1}]$¤ fl$_1$ = 7g$[x_t > x_{t+1}]$¤ fl$_2$ = 3g searches for a week-long increase followed by a 3 days decrease in values of x. The braces are necessary, because the use of $[l_1 = 7]$ and $[l_2 = 3]$ would return a match of 12 data points instead of the correct length of 10

The pattern description language represents a tool for feature extraction of time series. For example the classical technical trading methods based on indicators like moving average, RSI or momentum indicators can easily be expressed as a condition. Chartist methods aimed at finding graphical patterns in price plots (head & shoulders, spikes, flags etc.) require more sophisticated pattern descriptions but, their stylized shapes can also be described by regular expressions (a toy example of a simple pattern description of this type is given in Section 5). An interesting characteristic of the PDL comes from the fact that the length of a pattern has not to be specified ex ante. Perhaps the most commonly used tools for extracting information from time series work with data vectors of fixed length. Kernel regression, nearest-neighbor estimators and neural networks are all examples of procedures that "learn" output values from features described by input data composed of fixed number of observations. On the other hand, a regular expression can identify patterns without knowing at designing time the number of observations that compose it. In this manner, the same regular expression can capture qualitatively identical phenomena, which unfold on different time scales (fractal patterns) or stretch over time windows of variable length. Furthermore, with regular expressions we can test functional and/or statistical relationships in multi-dimensional data. Consider e.g. the simple condition $[x_{3t} = \sin(x_{2t})$¤$x_{1t}]$. It recognizes fragments of a multivariate time series, in which the exact, functional relation between $x_1; x_2$ and x_3 holds. On the other hand, [j x_t ¡ $a_1x_{t¡1}$ ¡ $a_2x_{t¡2}$ j< "], will match the data from an AR(2) process, whenever no realization of the innovations exceeds ".

4. The pattern matching machine

This section describes briefly, how the algorithm for patter recognition works. This algorithm is based on a pattern matching machine for text search as described for example in [7]. First a regular expression that encodes a pattern is compiled to a nondeterministic pattern matching machine (PMM). The latter is implemented as a finite state automaton (FSA). Essentially, every state of the FSA represents a condition or an operator ("+" or "*") that are found in the pattern description. The machine has a unique initial state and a unique final state. When started out in the initial state, the machine should be able to recognize any fragment of the matrix x, satisfying the pattern description, by reading values of x indexed by functions of t and changing state according to its rules, ending up in the final state. What makes the machine nondeterministic are some states which can point to different successor states (for example, the state which corresponds to the "or"-operator points to 2 successor states that check the expressions in the left and right operand, respectively). Intuitively, the matching algorithm works as follows: the PMM constructed moves forward along the time axis in discrete time steps, verifying at each step the condition encoded in its current internal state and changing this state according to the result of the verification. This result depends on the position in the matrix x (indicated by the time index t), where the verification takes place. The PMM can travel from a state A to a state B, pointed to by state A, whenever the evaluation of the state B yields the value true. If no such state exists, the machine returns to the initial state. If a state that points to the final state is eventually reached, a match has been found.

5. An application

The PDL developed in the previous sections can be applied to a variety of areas. Here an example of an application to technical analysis of price data will be given. A wide variety of theoretical and empirical models have been proposed to explain why technical trading is widespread in financial markets (cp. for instance references in [9] and [2]). Without entering the discussion between chartists and fundamentalists, we will search for patterns that are supposed to show up in plots of prices and are used by chartist community to predict the future behavior of markets. Consider, for instance, the well-known chartist technique of calculation of price targets based on the "golden ratio" [6]. In figure 1 the price target P3 is given as

$P3 = P2 \text{ ¡ } (P2 \text{ ¡ } P1) \text{ ¤ } 2:618$ (the constant 2.618 is closely related to the "golden ratio" $\frac{\text{p}\overline{5}+1}{2}$ ¼ 1:618).

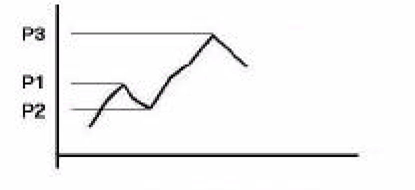

Figure 1. The 2.618 calculation

With the pattern description P, defined below, we will look for sequences (of indefinite length) of observations from New Yorker Stock Exchange that verify the stylized version of price targeting as shown in Fig.1:

$P = [x_t < x_{t+1}] \text{ ¤ } [x_t > x_{t+1}] \text{ ¤ } [x_t < x_{t+1}] \text{ ¤ } \text{f}x_t \text{ ¸ } x_{t+1}\text{gf}((x_{t_0+l_1+l_2} + (x_{t_0+l_1} \text{ ¡ } x_{t_0+l_1+l_2}) \text{¤} 2:618) \text{ ¼ } x_t) \wedge l_1 > l_2 \wedge l_3 > l_2\text{g}$

The first part of this pattern definition, $[x_t < x_{t+1}] \text{ ¤ } [x_t > x_{t+1}] \text{ ¤ } [x_t < x_{t+1}] \text{ ¤ } \text{f}x_t \text{ ¸ } x_{t+1}\text{g}$, stands simply for "rising followed by falling followed by longest, rising sequence of data points" ($\text{f}x_t > x_{t+1}\text{g}$ marks the end of the last rising sequence). The second part (composed of the last expression in braces) is decisive in selecting only those fragments of data that satisfy the "golden ratio rule". Here, t_0 is the time index of the first, t, of the last observation in a fragment of data matched, "¼" is a user defined operator meaning "approximately equal" and $l_1; l_2; l_3$ contain the lengths of the three sequences defined in the first part. Consequently, $x_{t_0+l_1}$ stands for P1, $x_{t_0+l_1+l_2}$ for P2 and x_t ($= x_{t_0+l_1+l_2+l_3}$) for P3. $(x_{t_0+l_1+l_2} + (x_{t_0+l_1} \text{ ¡ } x_{t_0+l_1+l_2}) \text{ ¤ } 2:618) \text{ ¼ } x_t$ checks, if the price target has been (approximately) reached. Conditions $l_1 > l_2$ and $l_3 > l_2$ specify that the two rising sequences have to be longer than the falling sequence between them and capture the typical shape of a rising market.

Fig. 2 shows matches of the pattern P that have been found in the data from NYSE between 15 October 1997 and 10 February 1998 (80 data points). The two first matches are 8 and the last match 7 observations long[3]

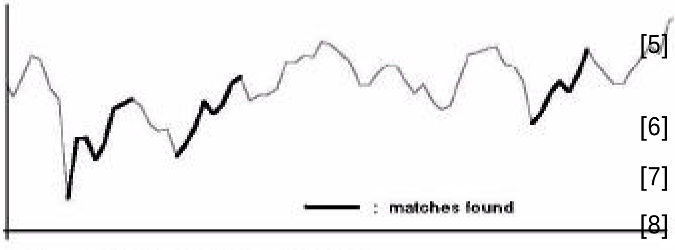

Figure 2. NYSE 15.10.1997 - 10.02.1998

6. Conclusions and further research

Based on the algorithm for pattern matching in a character string, a pattern description language has been developed. The compilation of a regular expression, that conforms to the PDL, creates a PMM that can be used as a searching device for detecting patterns or functional (statistical) relationships in multidimensional data .

This is just the first step on the way to isolate "niches of predictability" in data streams. It remains the non trivial task of finding patterns that, at the same time, can be formulated as a regular expression and are good predictors of something. The further work will therefore concentrate on two issues: First, on encoding of empirically observed patterns (e.g. the patterns commonly used in technical analysis) and verifying their predictive power. Secondly, it is planned to use GA to build up regular expressions and to run evolutionary selection in order to determine the fittest among them. Regular expressions are just strings of symbols that conform to a simple grammar. A GA, like the one used in [8], can build up such expressions from building blocks and then break up the strings again. The natural selection process causes suitable blocks to survive and combine among themselves while useless parts disappear. The "usefulness" or fitness of a pattern can be assessed, for instance, by the variance of a time series, calculated in a suitable chosen time window after the occurrence of that pattern.

References

[1] Agraval, R., Srikant, R., Mining Sequential Patterns, In Proc. of the 11th Intl. Conf on Data Engineering, ICDE 1995

[2] LeBaron, B., Arthur, W.B., Palmer, R., 1999, Time series properties of an artificial stock market, Journal of Ec. Dynamics & Control, 23,1487-1516.

[3] Garafalakis, M., Rastogi, R., Shim, K., SPIRIT: Sequential Pattern Mining with Regular Expression Constraints, In Proc. of the 25th VLDB Conference, Edinburgh, Scotland, 1999

[4] Mitchell, M., An Introduction to Genetic Algorithms, MIT Press 1998

[5] Packard, N. 1990, A Genetic Learning Algorithm for the Analysis of Complex Data, Complex Systems 4, No. 5, 543-572

[6] Plummer, T., Forecasting Fin. Markets, Kogan Page 1998

[7] Sedgewick, R., Algorithms, Addison Wesley 1988

[8] Szpiro, G., 1997, A Search for Hidden Relationships: Data Mining with Genetic Algorithms, Computational Economics, 10: 267-277

[9] Wolberg, J.R., Expert Trading Systems, John Wiley & Sons, 2000

[3] Running the search for 500 data points from NYSE resulted in 11 matches with lengths ranging between 6 and 19 observations