Discovering Patterns in Real-Valued Time Series

Joe Catalano, Tom Armstrong, and Tim Oates

University of Maryland Baltimore County
Baltimore, MD 21250 USA
{jcat1, arm1, oates}@umbc.edu

Abstract. This paper describes an algorithm for discovering variable length patterns in real-valued time series. In contrast to most existing pattern discovery algorithms, ours does not first discretize the data, runs in linear time, and requires constant memory. These properties are obtained by sampling the data stream rather than processing all of the data. Empirical results show that the algorithm performs well on both synthetic and real data when compared to an exhaustive algorithm.

1 Introduction

Many of the data generated and stored by science, government, and industry are multi-variate, real-valued, and streaming. These time series data come from such diverse sources as financial markets, climatological satellites, and medical devices. The potential uses of time series data are as varied as their sources. In some cases, the goal is to make accurate predictions (e.g., finding patterns in the fluctuations of the price of one stock that predict the price of another stock 3 days hence). In other cases, the goal is to gain a deeper understanding of the underlying system generating the data. This paper is concerned with the latter task, and presents a novel algorithm for finding recurring patterns (sometimes called motifs [1]) in time series.

Most algorithms for discovering patterns in time series have one or more of the following characteristics. The most common characteristic is an inability to work with real-valued data except through prior discretization [2,3]. Even in those cases where real-valued data are acceptable, multi-variate data typically are not [4]. The algorithms also tend to be batch [5], rather than incremental, which poses problems when the datasets are large or come from a high-volume stream. Finally, there are often assumptions about the number or temporal extent of patterns that exist in the data [6]. In contrast, we present an incremental algorithm with linear time and constant memory requirements for finding recurring patterns in real-valued, multi-variate streaming data wherein the number and temporal extent of the patterns are not known in advance.

The algorithm obtains these desirable properties by sampling. Given a data stream, a user-specified number of large *candidate windows* of data are sampled from the stream, sub-windowed, and stored. As time progresses, *comparison windows* of the same size are periodically sampled, sub-windowed, compared to all of the sub-windows from each of the candidate windows, and finally discarded.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 462–469, 2006.

The similarity scores (according to DTW) of the k most similar comparison sub-windows are kept for each candidate sub-window. If a candidate window contains an instance of a pattern, then some of its sub-windows will contain parts of that pattern, and the sampling process will yield comparison windows and sub-windows with the same properties. To distinguish patterns from noise, the mean similarity of the best matches for each candidate sub-window is compared to a distribution of mean similarities constructed so as to enforce the null hypothesis that the matching sub-windows do not contain instances of the same pattern. When the null hypothesis is rejected, the candidate sub-window probably contains a part of a pattern instance, and adjacent sub-windows with this property are "stitched" together to obtain the entire pattern.

Empirical results with a variety of datasets are reported. First, synthetic datasets in which known patterns are added to a background of noise are used to explore the ability of the algorithm to discover patterns of varying lengths and frequencies of occurrence in the face of changes to user-defined parameters. Second, the algorithm is run on real datasets for which the "true" patterns are not known. The discovered patterns are compared to those found by an exceptionally expensive algorithm that performs an exhaustive search for patterns of all lengths in the data. Results show that our algorithm finds many of the same patterns found by the exhaustive algorithm, but with limited computation and memory.

The remainder of this paper is organized as follows. Section 2 describes approaches to discovering patterns in data through discretization. Section 3 presents our sampling algorithm and complexity analysis. Section 4 contains empirical results of our algorithm on real-valued data. Finally, section 5 summarizes our contribution and points to future research directions.

2 Related Work

Extensive work has been done in the area of time series data mining, but little of it has focused on mining recurring patterns in real-valued time series. Some have applied clustering techniques to time series to mine interesting features of the data.

Dynamic Time Warping (DTW) is used in Oates et al. [7] to cluster the experiences of a mobile robot, using robotic sensor data as the source of the time series, and in [8] to cluster multi-variate real-valued time series produced by selecting one of k HMMs. While not focused on pattern discovery, they establish that DTW can reliably be used as a similarity measure of real-valued multivariate data.

Colomer et al. [2] use DTW to classify patterns belonging to the same class of operating situations in a level control system. Unlike our approach, they apply DTW to *episodes* rather than to the original time series. Keogh and Pazzani [3] propose a modification to DTW that operates on a piecewise linear representation of the data. Again this differs from our approach as it does not operate on the raw observations.

Lin et al. [4] define a *motif* as a previously unknown frequently occurring pattern, and introduce a discovery method that uses a modified Euclidean distance function to improve the complexity of the search. To achieve this performance they must reduce the dimensionality of the time series by means of a piecewise aggregate approximation and then further transform the time series into a discrete representation. Chiu et al. [1] extend Lin's work, addressing the scalability of the motif discovery algorithm and its ability to discover motifs when noise is present in the time series. Lin et al. [9] extend the symbolic representation introduced in [4] to better work with algorithms for streaming time series.

Finally, Oates [5] investigates a variation of the pattern discovery problem, wherein they determine what makes a set of sequences different from other sequences obtained from the same source. The approach operates on multi-variate real-valued time series and uses DTW as the similarity measure, but does not use a sampling approach.

3 Algorithm

Our sampling algorithm, figure 1, accomplishes two main tasks; the first is to discover windows from a time series that contain pattern instances and the second is to remove noise prefixes and suffixes in windows containing patterns. It accomplishes these goals by repeatedly sampling fixed windows of the time series looking for pairs of windows that have a small distance under DTW. We use the distance between the windows as a measure of similarity, with large distances indicating dissimilarity. We further constrain the algorithm by requiring it to discover patterns without a priori knowledge of their existence. We must then define some threshold for rejecting a match when the DTW distance between the windows is too large. Since the goal is to distinguish patterns from noise, a distribution of DTW distances between noisy windows is computed and used as a reference for thresholding the quality of a match. The algorithm performs these tasks using bounded time and memory by fixing the number and size of the windows that are compared. For an incremental version we sample windows on demand.

3.1 Noise Distribution

The sampling algorithm has no a priori knowledge of the existence of patterns in the time series. We define the null hypothesis to be that two randomly sampled windows do not contain instances of the same pattern. A distribution of window similarities must be computed as a basis for rejecting the null hypothesis when two windows contain a pattern instance. This is problematic because the core assumption is that given a large enough window of time series, it will contain some or all of a pattern instance with high probability if the patterns occur frequently. We create windows containing non-contiguous observations from the time series which ensures that these windows contain a pattern instance with low probability. That is, we create a noise window of length w by randomly sampling

and concatenating w individual observations. Warping these *noise windows* with normal windows gives us a null hypothesis distribution.

3.2 Pattern Discovery

In the pattern discovery phase of the algorithm, we repeatedly sample *candidate windows* from the time series and store the k-best matching *comparison windows* to each. Our goal is to identify frequently occurring patterns, thus we rely on sampling a sufficient number of windows to increase the probability of capturing multiple instances of a pattern. For the algorithm to be successful, the window length we choose must be large enough to fully contain any pattern we expect to find, but this also means the windows will contain noise as well. Having noise in the windows will increase the distance between them making it difficult to identify legitimate patterns. This is in addition to the problem of extracting only the pattern from the window. To address both of these problems we consider *sub-windows*. The relatively small size of a sub-window makes it useful for addressing this issue of granularity. Large windows contain noise and a large ratio of noise to pattern will yield poor results under any distance measure because the percentage of the window that will not have a strong matching region in another window will be great.

The patterns are discovered as follows. Create two sets of sub-windows, the *candidate set*, denoted *candSW*, and the *comparison set*, denoted *compSW*. It is the candidate set from which we reject sub-windows that come from the noise distribution and identify patterns from the remaining sub-windows. To populate these sets, sample a window W having length w and generate all sub-windows W_i having length $\bar{w}$. This yields $(w - \bar{w}) + 1$ sub-windows which are added to either set (*candSW* or *compSW*) and repeat this process to a specified bound. Normalize each sub-window to have mean 0 and standard deviation 1. When both sets are populated, apply DTW to all pairs of sub-windows between the two sets. Group the sub-windows in *compSW* by the window from which they were created and add only the best W_i in *compSW* from each group to the list of matches for W_i to which it is being compared. After warping all pairs, reduce the list of matches for each W_i in *candSW* to the k with the smallest distance measures.

In the final step of the algorithm, bad matches are removed from the candidate set. Recall the noise distribution that was created by warping normal sub-windows to sub-windows containing pure noise. As with candidate sub-windows, keep only the k-best matches for each noise sub-window. Sort the noise set by increasing average distance and a rejection threshold, γ, is established. Given some α, $\gamma = \lfloor(\alpha * n)\rfloor^{th}$ average warp distance where n is the number of noise sub-windows. Reject those sub-windows that have an average warp distance greater than γ on the basis that the observed value is common in the noise distribution and therefore is not likely to be a pattern instance. After removing the sub-windows containing bad matches, repeat the entire process using the now smaller candidate set, the same noise set, and a new comparison set. The process of eliminating candidate sub-windows is inherently error-prone because the comparison windows are randomly chosen and therefore we may not get enough

windows containing patterns to accurately accept or reject the candidate sub-windows. By running multiple iterations of the algorithm we reduce the amount of error introduced into the process.

```
SAMPLING(timeSeries, w, w̄, iterations, alpha)
 1  amtToSample ← AMOUNTTOSAMPLE(timeSeries, w, 50)
 2  compSW ← CREATESWSET(timeSeries, w, w̄, amtToSample)
 3  candSW ← CREATESWSET(timeSeries, w, w̄, amtToSample)
 4  noiseSW ← CREATENONCONTIGUOUSSWSET(timeSeries, w)
 5  combSW ← {∅}
 6  ADDALL(noiseSW, combSW)
 7  ADDALL(candSW, combSW)
 8  for i ← 1 to iterations
 9       do
10          if (iterations > 1)
11            then
12                 compSW ← CREATESWSET(timeSeries, w, w̄, amtToSample)
13                 COMPAREALLSUBWINDOWPAIRS(candSW, compSW)
14            else
15                 COMPAREALLSUBWINDOWPAIRS(combSW, compSW)
16
17          REMOVEREJECTS(alpha, candSW, noiseSW)
18
```

Fig. 1. Batch Mode Pattern Discovery

When all iterations have completed we then stitch together the remaining candidate sub-windows to form patterns. A pattern is formed by combining all overlapping and adjacent sub-windows. The resulting window is then considered to be a complete pattern instance.

The number of subwindows and the number of comparisons by DTW dominates the space and time complexity, respectively. DTW has quadratic time and space complexity, but the algorithm only considers pairs of subwindows of length $\bar{w}$, resulting in constant time and space costs. The algorithm samples n candidate windows of length w and m comparison windows of length w requiring $\mathrm{O}(n)$ and $\mathrm{O}(m)$ time and space. Each window of length w has $(w - \bar{w} + 1)$ subwindows of length $\bar{w}$. Therefore, there are a total of $n * m * (w - \bar{w} + 1)^2$ subwindows to compare and store, or $\mathrm{O}(nm)$ time and space. When considering the incremental version of the algorithm, $m = 1$ so the complexity is linear in the number of candidate windows.

4 Experiments

We have evaluated our algorithm by studying performance on a synthesized time series with a recurring, embedded noisy pattern. Then we explore the uni-variate time series of the Standard & Poor's 500.

We generated a uni-variate time series of 10,000 observations sampled uniformly in the range $[0, 10]$. The first 18 observations were used as a template for the pattern to be embedded in the data. The template is duplicated 49 times and noise chosen uniformly in the range $[-1.0, 1.0]$ is added to each observation in every copy of the template. We then insert the copies into the time series starting at position 100 and incrementing by 100 until all 49 have been embedded.

We ran the sampling algorithm on the synthetic data varying the values for $\bar{w}$ and α and we fixed w as $10*\bar{w}$. To $\bar{w}$ we assigned values from the set $\{5, 10, 15, 20\}$ and to α values from the set $\{0.05, 0.10, 0.15, 0.20\}$. It is expected that the noise distributions will be normally distributed, and the false-positive rate should increase proportionately with greater values of α. Sub-windows that overlap a pattern should have higher quality matches than those that do not overlap a pattern, and when sub-window stitching is performed the stitched pattern should be 18 observations in length and align with the start of an embedded pattern. Running multiple iterations of the sampling algorithm for a given set of parameters, α and $\bar{w}$, should reduce the number of errors made when selecting candidate sub-windows to keep.

Type I and Type II errors made by the algorithm when it evaluates a sub-window for either acceptance or rejection are important statistical measures when determining if the algorithm is performing correctly. For our sampling algorithm, a Type I error is made when the *candidate* sub-window spans an instance of the embedded pattern but is rejected. Likewise, a Type II error is made when the *candidate* sub-window does not span an instance of the embedded pattern but is accepted. In this experiment we know the locations of every instance of the pattern. Therefore counting the exact number of Type I and Type II errors is possible.

Table 1. Type I & II Errors, $\bar{w} = 5$, $\alpha = 0.05$

	Iteration	Total	Accept	Error Rate
Pattern	1	209	105	49.7%
Pattern	10	43	42	2.3%
Pattern	25	27	26	3.7%
Not Pattern	1	1057	45	4.2%
Not Pattern	10	5	4	80%
Not Pattern	25	2	2	100%

Table 1 displays the totals, categorized as sub-windows that span only pattern observations and sub-windows that span only noise observations, after one, ten, and twenty five iterations of the sampling algorithm for $\bar{w} = 5$ and $\alpha = 0.05$. The data are a good indication that our noise distribution is an accurate model of the noise contained in the time series and performs well at eliminating sub-windows spanning noise. After 25 iterations, we have nearly eliminated all sub-windows spanning noise while keeping the percentage of Type I errors to a minimum. For different values of α (i.e. 0.10 and 0.20), the algorithm performed similarly.

4.1 Standard and Poor's 500

In our real dataset experiment, we ran our algorithm on the daily closing price of the S&P 500 for approximately 33 years. The data were 8760 uni-variate time series observations, each fell in the range $[4.4, 49.74]$. As was the case with the synthetic experiments, the parameters for this experiment were $\alpha = \{0.05, 0.10, 0.15, 0.20\}$ and $\bar{w} = \{5, 10, 15, 20\}$.

Unlike in the previous experiment, we did not have knowledge of any patterns occurring in this data. This made it necessary to perform an exhaustive search of the time series to produce a basis for evaluating the quality of the patterns discovered by our algorithm. Likewise, it is impossible to collect statistics with regard to the quantity of Type I and II errors that were made, because there is no way to determine when a window contains pattern observations and when it does not.

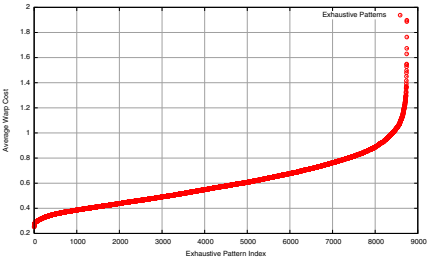

Fig. 2. S&P 500 Exhaustive Costs $\bar{w} = 10$

Figure 2 shows the patterns found exhaustively as a function of their average warp cost and sorted from best to worst. The exhaustive search warped all pairs of sub-windows for a given sub-window length (ignoring overlapping sub-windows) and maintained a list of the ten best matches for each candidate. The warp cost depicted in the figures are the average of those ten matches for each candidate sub-window. The plot shows that a small number of the candidates had exceptionally good matches and a small number had exceptionally poor matches.

The top-ten candidate sub-windows found using the sampling algorithm for sub-window size 10 all were in the 93rd-percentile out of 8751 sub-windows under the exhaustive search results. Three of the ten results were in the top-100 candidates in the exhaustive search results. We have experimented on additional datasets, including commonly cited multi-variate datasets (e.g. winding, subcutaneous). The results of our algorithm on these data are analogous to the uni-variate cases.

5 Conclusion

This paper described an incremental algorithm with linear time and constant memory requirements for finding recurring patterns in real-valued, multi-variate

streaming data wherein the number and temporal extent of the patterns is not known in advance. Empirical results with synthetic data showed that it successfully finds known patterns and is robust with respect to the settings of user-specified parameters. Empirical results with real data show that the patterns found by the algorithm compare favorably to an (impractically expensive) exhaustive algorithm. Future work will focus on applications with high-volume data streams (e.g., audio data) and automated representation tuning by, for example, searching over sets of basis functions for those that yield high quality (low match cost) patterns.

References

1. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD'03). (2003) 493–498
2. Colomer, J., Melendez, J., Gamero, F.I.: Pattern recognition based on episodes and DTW, applications to diagnosis of a level control system. In: Proceedings of the Sixteenth International Workshop on Qualitative Reasoning. (2002)
3. Keogh, E., Pazzani, M.: Scaling up dynamic time warping to massive datasets. In Zytkow, J.M., Rauch, J., eds.: 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99). Volume 1704., Prague, Czech Republic, Springer (1999) 1–11
4. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proceedings of the Second Workshop on Temporal Data Mining, Edmonton, Alberta, Canada (2002)
5. Oates, T.: Identifying distinctive subsequences in multivariate time series by clustering. In Chaudhuri, S., Madigan, D., eds.: Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, ACM Press (1999) 322–326
6. Bozkaya, T., Yazdani, N., Ozsoyoglu, Z.M.: Matching and indexing sequences of different lengths. In: CIKM. (1997) 128–135
7. Oates, T., Schmill, M.D., Cohen, P.R.: A method for clustering the experiences of a mobile robot that accords with human judgments. In: AAAI/IAAI. (2000) 846–851
8. Oates, T., Firoiu, L., Cohen, P.R.: Clustering time series with hidden markov models and dynamic time warping (1999)
9. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the Eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, CA, USA (2002)