Information Sciences 450 (2018) 73–88

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Fast multi-subsequence monitoring on streaming time-series based on Forward-propagation

Xueyuan Gong*, Simon Fong, Yain-Whar Si

*Department of Computer and Information Science, University of Macau, Macau, China*

ARTICLE INFO

*Article history:*
Received 22 September 2017
Revised 8 March 2018
Accepted 9 March 2018
Available online 11 March 2018

*Keywords:*
Streaming time-series
Subsequence monitoring
SPRING
NSPRING
FPNS
DTW

ABSTRACT

Streaming time-series has drawn unprecedented interests from the computer science researchers. It requires faster execution time and less memory space than traditional approaches in processing historical time-series. Given the real-time constraint in the analysis over streaming time-series, a proper pre-processing step may not even be applicable. Subsequence monitoring is one of the main functions used in a wide range of time series related applications, e.g. quantitative trading in the stock market. In this paper, we propose a novel approach for multi- subsequence monitoring on streaming time-series. The proposed Forward-propagation NSPRING (FPNS) approach is inspired by the forward propagation mechanism in Artificial Neural Networks (ANN). In our proposed approach the concept of forward propagation is adopted to by-pass the unnecessary calculations as in NSPRING where the whole matrix is computed for the final result. FPNS computes a small part of the matrix by indexing only the necessary calculations with the aid of the forward propagation mechanism. As a result, FPNS can effectively reduce the execution time. In the experiments, we compared the scalability, execution time and memory requirement of FPNS, NSPRING, and UCR-DTW using synthetic and real datasets. The experimental results show that on average, FPNS is about three times faster than NSPRING and one order of magnitude faster than UCR-DTW. In addition, FPNS preserves the same accuracy with NSPRING while FPNS runs much faster than NSPRING.



## 1. Introduction

Increasing number of real-time applications generate vast number of streaming time-series every day. These time-series comprise of data from computer networks, real-time traffic, electrocardiogram (ECG), electroencephalogram (EEG), stock price, and weather information. In contrast to traditional historical time-series, updates over streaming time-series happen in real-time. Such real-time property poses a significant challenge for data mining researchers in designing efficient algorithms for processing streaming time-series. In particular, streaming time-series online is often fast. The fast updates demand for lightweight algorithms that are designed to execute very quickly, using less memory space, and perhaps without any data preprocessing. In recent years, many techniques on the processing of streaming time-series were proposed. These techniques include classification [11,19], clustering [26,30], rule discovery [25], motif discovery [29] and subsequence monitoring [4,20,23].

* Corresponding author.
*E-mail addresses:* yb47453@umac.mo, amoonfana@qq.com (X. Gong), ccfong@umac.mo (S. Fong), fstasp@umac.mo (Y.-W. Si).

https://doi.org/10.1016/j.ins.2018.03.023
0020-0255/

Among all the processing techniques on streaming time-series, subsequence monitoring is considered one of the most fundamental methods since it is widely used in a number of real-world applications, such as technical pattern monitoring in stock market trends. A technical pattern is thought to be a special subsequence in the stock time-series, which can be used as signal indicators for the upward/downward direction of the stock price movement. Therefore, monitoring technical patterns in stock markets helps traders and analyzers to predict the forthcoming stock market trends. Given a sequence (a time-series), the objective of subsequence monitoring is to identify the subsequences which are the most similar to the query sequence in real-time. In streaming time-series, the demand on the efficiency of the subsequence monitoring method is significantly higher than other types of time series processing because streaming time-series cannot be preloaded and preprocessed from some static data archive. In 2007, Sakurai et al. [23] proposed a method called SPRING to monitor subsequences in streaming time-series. SPRING is faster than the naive method by five orders of magnitude without sacrificing accuracy. It was reported that the time complexity of SPRING is O($nm$), where $n$ is the length of sequence and $m$ is the length of query sequence. However, SPRING does not support normalization which is prerequisite for obtaining meaningful results from some of the data sets [8,20]. To alleviate this problem, Gong et al. [4] proposed an algorithm called Normalization-supported SPRING (NSPRING) to support normalization while maintaining the temporal and space complexity of SPRING.

However, in multi-subsequence monitoring problems, instead of monitoring just one subsequence, thousands of subsequences are required to be monitored at the same time. In such cases, an algorithm which is much faster than the currently available one would be desirable. For example, a quantitative trading system may need to monitor thousands of stock prices simultaneously for detecting technical patterns. In such circumstances, the system needs to react in seconds or even in milliseconds as stock prices change at any moment. Any delayed reaction may miss buying or selling opportunities. To address these problems, Forward-propagation NSPRING (FPNS) algorithm for multi-subsequence monitoring is proposed in this paper. The proposed method is inspired by the forward propagation mechanism in Artificial Neural Network (ANN). Instead of using propagation for tuning parameters in NN, it is used for indexing necessary calculations in the FPNS thereby excluding all unnecessary calculations to improve the execution time. Specifically, FPNS needs to calculate a $n$ by $m$ matrix. Among the matrix, necessary cells in the matrix are indexed and calculated while all unnecessary ones are ignored. In some extreme scenarios, the time complexity of FPNS is O($nm$). In normal cases, FPNS can effectively prune at least half of the calculations.

A similar idea is also used in the method called UCR-DTW proposed by Rakthanmanon et al. [20]. UCR-DTW does not reduce the time complexity of original algorithm (i.e. O($nm^2$)). However, it aims to accelerate the Dynamic Time Warping (DTW) distance calculation by pruning unnecessary computations. Interestingly, UCR-DTW takes only 34 h to run on the sequence of length trillions (e.g. 1,000,000,000,000). Thus, pruning unnecessary calculations can significantly improve the algorithm's performance.

In this paper, we first compare the scalability of FPNS with NSPRING and UCR-DTW on synthetic datasets. Next, FPNS, NSPRING and UCR-DTW are tested on the same benchmark datasets from UCR archive [2]. Finally, FPNS, NSPRING and UCR-DTW are compared on UCI repository [12] to validate their capability on multi-variate sequences. From the experiment results, we find that FPNS is three times faster than NSPRING and one order of magnitude faster than UCR-DTW. The time complexity of FPNS and NSPRING are both linear (i.e. O($lnm$), where $l$ is the number of query sequence, $n$ is the length of time-series and $m$ is the length of query sequence). For UCR-DTW, we observe that the pruning ability of UCR-DTW depends on the convergence of minimum distance $D_{min}$ (Section 4). Specifically, the fast convergence rate of $D_{min}$ to a smaller value enables UCR-DTW to reduce as much calculations as possible. However, this property is less useful in multi-subsequence monitoring since $D_{\min}$ will converge from the initial state for each query sequence. In other words, the UCR-DTW prefers a query sequence and a time-series of length 1,000,000,000, rather than 1000 query sequences and a sequence of length 1,000,000. Thus, The scalability of UCR-DTW is quadratic based on time complexity O($lnm^2$).

We review the related work in Section 2. In Section 3, we first introduce the notations and definitions used in this paper. Next, the FPNS algorithm is described in details. Finally, the experiment results are discussed in Section 4 while Section 5 concludes the paper.

## 2. Related work

A number of applications and relevant methods on streaming time-series were proposed in recent years. In multi-dimensional streaming time-series classification area, Hu et al. [6] proposed a novel framework to increase the accuracy by considering weights on the class prediction from each stream. Do et al. [3] proposed a Multi-modal and Multi-scale Temporal Metric Learning (M2TML) framework for a robust kNN classifier. Wan and Si [27] formally defined 53 chart patterns in financial time-series for the classification of them. In approximation area, Luo et al. [13] proposed a real-time algorithm that generates the optimal Piecewise Linear Approximation (PLA) in terms of representation size while maintaining the max-error guarantee on streaming time-series. In correlation discovery area, Guo et al. [5] proposed a framework called AEGIS, which exploits statistical properties to prune the calculation on time-series pairs to speed up the correlation discovery process. In pattern detection area, Chen et al. [1] proposed a novel distance measure of time-series namely Spatial Assembling Distance (SpADe), for pattern detection on streaming time-series. The SpADe can handle noise, shifting, and scaling in both temporal and amplitude dimensions. Miao et al. [14] presented a novel pattern detection method, which is based on the notions of templates, landmarks, constraints and trust regions instead of accounting for temporal and magnitude deformations. These approaches show the extent of challenging problems that researchers encounter when addressing various applications on streaming time-series.