Optimal Segmented Linear Regression for Financial Time Series Segmentation

Chi-Jen Wu
Department of CSIE
Chang Gung University
Taiwan
cjwu@mail.cgu.edu.tw

Wei-Sheng Zeng
Institute of Information Science
Academia Sinica
Taiwan
wszeng@iis.sinica.edu.tw

Jan-Ming Ho
Institute of Information Science
Academia Sinica
Taiwan
hoho@iis.sinica.edu.tw

***Abstract*—Given a financial time series dataset, one of the most fundamental and interesting challenges is the need to learn the stock dynamics signals in the financial time series data. An essential task is to represent the time series in line segments which is often used as a pre-processing step for learning the marketing signal patterns in financial computing. In this paper, we focus on the optimization problem of computing the best segmentations of such time series based on segmented linear regression models. The major contribution of this paper is to define the problem of Multi-Segment Linear Regression (MSLR) of computing the optimal segmentation of a financial time series, denoted as the MSLR problem, such that the global square error of segmented linear regression is minimized. We present an optimum algorithm named OMSLR, with two-level dynamic programming (DP) design, and show the optimality of OMSLR algorithm. The two-level DP design of OMSLR algorithm can mitigate the complexity of searching the best trading strategies in financial markets. It runs in O(kn^2) time, where n is the length of the time series sequence and k is the number of non-overlapping segments that cover all data points.**

***Index Terms*—time-series, segmentation, segmented linear regression, signal learning and processing, piecewise polynomial representation.**

I. Introduction

Learning investment or trading signals from financial market data is one of most fundamental and interesting research challenges in both academia and industry [1], [2]. For example, the American hedge fund manager, Renaissance Technologies has leveraged financial signal processing technologies in stock trading for a long time. However, financial time series is difficult to summarize or to be represented due to its highly non-stationary nature [3]. For a given financial time series dataset, an initial processing step of learning the signal patterns is often to represent the time series in line segments to alleviate data uncertainty and noise [4].

In a segmentation process, a time series is divided into k non-overlapping segments, and each segment is represented by a model to describe data points in the approximation segments. The segment representation is measured using error functions depending on the requirements of their applications. Actually, time series segmentation is widely for dimensionality reduction purposes in economics [5], engineering [6] and science [7].

Time series segmentation has been extensively discussed in different domains and various models, which has resulted in many works such as the works in [8]. In 1961, the first version of time series segmentation problem is reported in [9] and a dynamic programming (DP) algorithm with time complexity O(kn^3) is also described. Time series segmentation also arises in data mining applications. The article [10] gives a review on applications of segmentation methods in data mining research area. The segmentation methods are classified into three categories, including sliding windows, bottom-up, and top-down methods. The experimental comparisons showed that the bottom-up method results in better performance than other methods [10].

In the past few years, only a few algorithms [11], [12], [13] have been proposed to reduce the time complexity of the time series segmentation problem. The objective is to simplify represent of large scales time series data. The Piece-wise Linear Approximation (PLA) [11] is a widely used approach for the segmentation task. Acharya et al., [13] presented near-linear time algorithms that achieve a significant improvement compared to the DP approach on large time series. The interested readers can refer to Esling and Agon [8] who present a survey on approximation segmentation of time series. To the best of our knowledge, the previous approaches have not addressed the problem of optimum segmentation of financial time series. Most of them discussed segmentation methods in terms of approximation representation [11], on-line processing [12] and their time complexity [13]. Also, most of the approaches mentioned are suboptimal. Due to the high computational complexity, optimal algorithms can not be easily applied to real-time applications.

In this paper, we are interested in the open question [14], how to best choose k, the optimal number of segments used to represent a particular time series. For financial trading strategies, k is a measure of number of times of changes in market trend. It is also an indicator of how many times to trade in the market while receiving a reasonable amount of trading profits [15]. Instead of answering the open question directly, we start with focusing on minimizing global square error for a given k, and also deriving the optimal representation of each of the k segments.

Firstly, we formulate the Multi-Segment Linear Regression (MSLR) problem and define the global square error as the performance index. Then, we present the Optimal Multi-Segment Linear Regression (OMSLR) algorithm, the two-level DP approach for producing the globally optimal segmentation. Finally, we show the optimality of the proposed OMSLR algorithm. The time complexity of the OMSLR algorithm is $O(kn^2)$, where n is the length of the time series and k is the number of non-overlapping segments that cover all data points. To the best of our knowledge, this work is the first to investigate the global optimal segmentation problem in time series processing, especially for financial time series.

This paper is organized as follows. We provide a brief description of the related work on time series segmentation in Section II. In Section III we present the formulation of segmentations as an optimization problem, named MSLR problem. In Section IV we present the OMSLR algorithm and show the optimality of the proposed OMSLR algorithm. The segmentation experiments are presented in Section V, and the results and future work are summarized in Section VI.

II. Related Work

In this section, the well known classical time series segmentation algorithms are described, and the performance analysis of the time complexity of these algorithms are also reported. Time series segmentation is an important direction in knowledge discovery research and has been extensively discussed from a data-mining perspective. There are at least two processing ways to represent a time series with straight lines, including linear interpolation approaches and line regression approaches [10]. The linear interpolation [16] that represents the data points by simply connecting the two endpoints of each segment and generates continuous line segments. And linear regression approaches that approximate a line segment by representing the best fitting line in the least square error and produce a set of non-overlapping segments [11]. However, the regression approaches result better in quality of the approximating segments generally [17]. Also, basic classification of these algorithms can be divided into two categories included on-line and off-line algorithms [18].

In this article we focus on the line regression approaches and off-line segmentation algorithms. The interested readers can refer to [19] for on-line methods of time series segmentation. The first group of algorithms is optimal solutions and the second one is heuristic. The main difference between two groups is heuristic algorithms are designed by obtaining the best time series segmentation such that the maximum error (global square error) for any segment is less than the user-specified threshold as a per-defined initial parameter. Most of existing segmentation solutions approximate segments based on the user-specified error. The optimal solutions are proposed to obtain the best segmentation with the minimum maximum error using given k segments. We discuss the classical heuristic algorithms first, and then the optimal solutions of time series segmentation are described.

Note that different fitness error functions should be used by the different approximate segmentation methods and applications, such as a regression, decision trees, and neural networks [20]. Depending on the application, the goal of the approximate segmentation is used to locate stable periods of time, to identify stock trading, or to simply represent the original time series into an approximation line. Our algorithm and assumptions are based on segmented liner regression models, the fitness error function in the present paper is measured by using the square error of each segment i.e., $\sum_{i=1}^{n}[y_i - \overline{y_i}]^2$. Thus, the naive computation complexity of the residual sum of squares over a given segment interval should take linear time $O(n)$.

A. Heuristic algorithms

Most popular of heuristic algorithms for time series segmentation usually can be classified into one of the following categories of algorithms [10]: 1) Bottom-up algorithm, 2) Top-down algorithm, 3) Sliding windows algorithm; unlike the previous two algorithms, it belongs to the category of the on-line algorithms, the detailed design of the Sliding windows algorithm should be referred to [10].

The Bottom–up algorithm is a greedy merge approach, and it is fast. Firstly, it starts by dividing the original time series, of length n, into $n-1$ segments. In this step, all the approximation errors of each segment are 0. Next, the pairs that cause the smallest increase in the fit error are consequently merged into a bigger segment based on the comparison of each pair of consecutive segments, including left and right segments. The algorithm iteratively merges the least fit error pair until the pre-defined stopping criteria is met, such as approximation error is less than the user-specified threshold. The time complexity of bottom–up algorithm is $O(Ln)$ [10], where n is the number of points of the time series and L is the average length of the representing segments (i.e., the average segment length is $L = n/k$).

The Top–down algorithm is also a greedy approach based on a divide and conquer design, it starts with the non-segmented time series (i.e., there is one major segment at initial step) and it introduces a new segment at each division step. In each iteration, it searches the best breaking point for placing the boundary to split the original segment into two segments based on the consideration of all possible breaking points. The process of division into two new segments is repeated until all the segments have approximation errors below the user-specified threshold. The division procedure is repeated without effect on the location of the breaking point determined in the previous iteration. The computational complexity is O(kn^2), where n is the number of points of the time series and k is the number of segments [10].

According to the literature [10], the Bottom–up algorithm can result in the lowest deviation in approximation error. Both methods are suboptimal approaches that may not be suitable for our problem.

B. Optimization algorithms

This optimal segmentation problem has been studied in 1961, the first version of time series segmentation problem is reported by Bellman [9] and a DP algorithm with time complexity O(kn^3) is presented [21]. Terzi and Tsaparas [22] presented a O(kn^2)-time implementation is possible for the constant segmentation problems. It is to be noted that an optimal solution of the DP algorithm with O(kn^2) time complexity is presented in [22], the fit error is measured by using the mean or median errors. Without a doubt this solution consists of a DP algorithm and can be computed in time O(kn^2). Guha et al., [23] presented a near optimal segmentation algorithm with provable error bounds, variations of this error bounded error segmentation problems have been studied extensively [24]. Another approach is to consider autoregression (AR) models based on placing prior distributions on the number of change points.

Compared to previous works, in this paper it is shown that a novel two-level DP design algorithm offers a direct approach to the determination of an optimal fit error (i.e., global square error) based on segmented liner regression models and results in the time complexity O(kn^2) for the time series segmentation problem. We have applied this algorithm to the financial time series segmentation for financial trading strategies. Researchers consider the financial time series segmentation problem in difference aspects, such as Chung et al., [25] proposed a genetic algorithm for this problem, a segmentation method based on the maximum or minimum turning points of a financial time series is presented in [26], and the clustering [27] representation approaches is also studied.

III. Formulation of Problem MSLR

A formal definition of the Multi-Segment Linear Regression (MSLR) problem is described in this section.

Given a time series $X = \{x_1, x_2, \ldots, x_n\}$ and an integer k, the objective to the MSLR problem is to partition X into k contiguous and non-overlapping segment intervals, i.e., $[l_{i-1}, l_i)$ and $[l_{k-1}, l_k]$, where $l_0 = 1, l_k = n, 1 \le l_i \le n, 1 \le i \le k-1, l_i \in \mathbf{N}$, such that the multi-segment linear regression square error, $\psi^2(1, n|\phi_k(X_n))$, with respect to the k-segment partition $\phi_k(X_n) = \{1, l_1, \ldots, l_k\}$ is minimized. Note that $\psi^2(1, n|\phi_k(X_n))$ is also denoted as the Global Square Error of the multi-segment linear regression representation, or GSE for short. In other words, we have

$$\begin{aligned} &\psi^2(X_n|\phi_k(X_n)) \\ &=\sum_{i=1}^{k-1} \sigma^2_{(l_{i-1}, l_i - 1)} + \sigma^2_{(l_{k-1}, l_k)} \\ &=\psi^2(X_{l_{k-1}-1}|\phi_{k-1}(X_{l_{k-1}-1})) + \sigma^2_{(l_{k-1}, l_k)}, \end{aligned} \tag{1}$$

where $\sigma^2_{i,j} = \sum_{m=1}^{j}(x_m - \mu(i,j,m))^2$ and $\mu(i,j,m)) = \beta_{i,j} \times m + \alpha_{i,j}$, $i \le m \le j$. More specifically, $\mu(i,j,m))$ is the linear regression model [28] of the time series X_n on the interval $[i, j]$ and $\sigma^2_{i,j}$ is the residual, also denoted as the square error, of linear regression. And $\alpha_{i,j}$ and $\beta_{i,j}$ are the linear regression parameters. Here $\alpha_{i,j}$, $\beta_{i,j}$ and $\sigma^2_{i,j}$ all can be computed in O(1) time using the following iterative equations, $\forall\ j,\ j \ge i$.

$$\alpha_{i,j} = \bar{x}_{i,j} - \beta_{i,j} \times \bar{t}_{i,j}, \tag{2}$$

$$\beta_{i,j} = \frac{\sum_{m=i}^{j}(x_m - \bar{x}_{i,j})(m - \bar{t}_{i,j})}{\sum_{m=i}^{j}(m - \bar{t}_{i,j})^2} \tag{3}$$

$$\begin{aligned} \sigma^2_{i,j} &= \sum_{m=1}^{j}(x_m - \mu(i,j,m))^2 \\ &= \sum_{m=1}^{j}(x_m - \beta_{i,j} \times m - \alpha_{i,j})^2, \end{aligned} \tag{4}$$

where $\bar{t}_{i,j} = \frac{(i+j)}{2}$, $\bar{x}_{i,j} = \frac{\sum_{m=i}^{j} x_m}{j-i+1}$, and $\mu(i,j,m) = \beta_{i,j} \times m + \alpha_{i,j}, i \le m \le j$. It can be shown that the above equations can be rewritten into iterative forms such that Algorithm 1 can be computed in O(n^2) time for all $1 \le i \le j \le n$. The detailed derivation of the equations, including Equation 2, 3 and 4 can be found in the Appendix A, thus we have the following Lemma 1.

Lemma 1. *The running time of Equation 2, Equation 3 and Equation 4 are all constant time O(1) for a time series $X = \{x_1, x_2, \ldots, x_n\}$.*

Therefor, we have the Lemma 2 as follows.

Lemma 2. *The running time of the Algorithm 1 is at most O(n^2) for a time series $X = \{x_1, x_2, \ldots, x_n\}$ and n is the number of data points of X_n.*

Proof. The proof is based on Lemma 1, and it is obviously that the time complexity of the Algorithm 1 is O(n^2), n is the length of a given time series. □

Given a fitness error function, the goal is to find the segmentation of the sequence and the corresponding representatives that minimize the fit error in the representation of the underlying data points. We call this problem a segmentation problem. As we mentioned before, in this paper the fit error is measured by using square error of each segment.

Algorithm 1 Matrix σ^2

```
Input:
    $X \leftarrow$ a time series data set
    $n \leftarrow$ the length of $X$
Initialize:
    $\alpha_{i,i} \leftarrow x_i$
    $\beta_{i,i} \leftarrow 0$
    $\sigma^2_{i,i} \leftarrow 0$
1:  /* for computing the matrix $\sigma^2$ */
2:  for $i = 1, 2, 3, \ldots n$ do
3:      for $j = i+1, \ldots n$ do
4:          $\bar{t}_{i,j+1} = \frac{(i+j+1)}{2}$
5:          $\bar{x}_{i,j+1} = \frac{(j-i+1)\bar{x}_{i,j} + x_{(j+1)}}{(j-i+2)}$
6:          $\alpha_{i,j} \leftarrow$ the derivation of Equation 2
7:          $\beta_{i,j} \leftarrow$ the derivation of Equation 3
8:          $\sigma^2_{i,j} \leftarrow$ the derivation of Equation 4
9:      end for
10: end for
Output: matrix $\sigma^2$
```

IV. OMSLR algorithm

We present the OMSLR algorithm for the MSLR problem as follows. Given a time series $X = \{x_1, x_2, \ldots, x_n\}$ and an integer k, the algorithm OMSLR iteratively segments the time series $X_j = \{x_1, x_2, \ldots, x_j\}$, where $1 \leq j \leq n$, into i segments, starting with $i = 1$ to $i = k$. Since Equation 1 is an iterative function, we can compute a matrix M to maintain the square error of possible segments by referring to the matrix σ^2. Based on the concept, we design a two-level DP Algorithm 2 to compute the matrix M, in which $M[i,j] = (\gamma_{i,j}, \rho^2_{i,j})$[1], for $i = 1 \to k$, as a representation of the best way of partitioning X_j into i segments $\forall\ j, 1 \leq j \leq n$. Here, $\gamma_{i,j}, 1 \leq \gamma_{i,j} \leq j$ denotes the starting point of the last segment of $\hat{\phi}_i(X_j)$ and the variable $\rho^2_{i,j}$ is the global square error of i-segment partition of X_j based on $\hat{\phi}_i(X_j)$.

$\gamma_{i,j}$ and $\rho^2_{i,j}$ can be computed by the following equations.

$$\begin{aligned} \gamma_{i,j} &= \arg \min_{(i-1)d < m \leq j-d} \{\rho^2_{(i-1),m} + \sigma^2_{(m+1),j}\}; \\ \rho^2_{i,j} &= \rho^2_{(i-1),(\gamma_{i,j}-1)} + \sigma^2_{\gamma_{(i,j)},j}. \end{aligned} \tag{5}$$

In Equation 5, the d is a constant value used to control the minimum size of a segmentation representation, the default value of d is 2.

With the matrix M and segmentation index $\gamma_{i,j}$, we can backtrack an i-segment partition on X_j denoted as $\hat{\phi}_i(X_j) = \{1, \hat{l}^{(i,j)}_1, \ldots, \hat{l}^{(i,j)}_i\}$, by the following equations.

$$\hat{l}^{(i,j)}_m = \begin{cases} j, m = i; \\ \gamma_{(m+1),(\hat{l}^{(i,j)}_{(m+1)}-1)}, 1 \leq m \leq i-1. \end{cases}$$

[1]Note that the pair of values $(\gamma_{i,j}, \rho^2_{i,j})$ stored in $M[i,j]$ is a tuple that is used to store multiple items in a single variable and also is a very common data structure in modern programming languages.

In specific, the k-segment partition of X_n, denoted as $\hat{\phi}_k(X_n) = \{1, \hat{l}_1, \ldots, \hat{l}_k\}$, is computed as follows.

$$\hat{l}_i = \begin{cases} n, i = k; \\ \gamma_{(i+1),(\hat{l}_{(i+1)}-1)}, 1 \leq i \leq k-1, \end{cases}$$

where $x_{\hat{l}_0} = x_1$ and $x_{\hat{l}_k} = x_n$. The algorithm OMSLR, given as Algorithm 2, provides an optimal solution for the MSLR problem. In the following, we show that $\hat{\phi}_k(X_n)$ is an optimal solution of the MSLR problem.

Theorem 1. *Given a time series $X_n = \{x_1, x_2, \ldots, x_n\}$ and the number of segments k, the i-segment partition $\hat{\phi}_{i+1}(X_j), \forall\ j, 1 \leq j \leq n, \forall\ i, 1 \leq i \leq k$ as computed by Algorithm OMSLR is optimum.*

Proof. We give a sketch of the proof and prove Theorem 1 by contradiction. We skip the case $k = 1$, it is a natural linear regression. For the case $k = 2$, it is obviously to see that $\forall\ j, 1 \leq j \leq n$, $\hat{\phi}_2(X_j)$ is optimum, since Algorithm OMSLR enumerated all the feasible solutions.

In the induction step, we assume that $\forall\ j, 1 \leq j \leq n$, $\hat{\phi}_i(X_j)$ is optimum. To show that $\hat{\phi}_{i+1}(X_j)$ is also optimum, $\forall\ j, 1 \leq j \leq n$, we assume that there exists an integer $\tau, 1 \leq \tau \leq n$, such that $\hat{\phi}_{i+1}(X_\tau)$ is not optimum. Let $\phi^*_{i+1}(X_\tau) = \{1, l^*_1, \ldots, l^*_i, l^*_{i+1} = \tau\}$ be the optimum $(i+1)$-segment partition of X_τ. Then we have the following equation:

$$\psi^2(X_\tau | \hat{\phi}_{i+1}(X_\tau)) > \psi^2(X_\tau | \phi^*_{i+1}(X_\tau)). \tag{6}$$

The induction assumption says that the i-segment partition, $\hat{\phi}_i(X_{l^*_i-1})$ is optimum, thus it implies the Equation 7:

$$\psi^2(X_{l^*_i-1} | \phi^*_i(X_{l^*_i-1})) = \psi^2(X_{l^*_i-1} | \hat{\phi}_i(X_{l^*_i-1})), \tag{7}$$

and Algorithm OMSLR also guarantees the Equation 8.

$$\begin{aligned} \psi^2(X_\tau | \hat{\phi}_{i+1}(X_\tau)) &\leq \psi^2(X_{l^*_i-1} | \hat{\phi}_i(X_{l^*_i-1})) + \sigma^2_{(l^*_i,\tau)} \\ &= \psi^2(X_{l^*_i-1} | \phi^*_i(X_{l^*_i-1})) + \sigma^2_{(l^*_i,\tau)} \\ &= \psi^2(X_\tau | \phi^*_{i+1}(X_\tau)) \end{aligned} \tag{8}$$

Equation 7 and 8 imply that the assumption of Equation 6 is a contradiction. Thus, we have Theorem 1. ☐

To analyze the computational complexity of algorithm OMSLR, the average case will be assumed. The time complexity of the algorithm OMSLR is O(kn^2). Next we give the detailed proof of Theorem 2.

Theorem 2. *The running time of the algorithm OMSLR is at most O(kn^2) for $X_n = \{x_1, x_2, \ldots, x_n\}$ and k is the number of non-overlapping segments of X_n.*

Proof. The following considerations will be taken into account for the proof of Theorem 2. All the points that are processed to obtain the matrix σ^2, it is required an order of complexity O(n^2) that shown in Lemma 2. In Algorithm 2, the computational complexity of processing each i-segment by backtracking the matrix σ^2 and maintaining

Algorithm 2 OMSLR(X, k, σ^2)

```
Input:
    $X \leftarrow$ a time series data set
    $k \leftarrow$ the number of segments
    $\sigma^2 \leftarrow$ the error matrix $\sigma^2$ form Algorithm 1
Initialize:
    $gse \leftarrow \emptyset$ /* the Global Square Error */
    $\gamma_{i,j} \leftarrow 0$
    $\rho_{i,j} \leftarrow 0$
    $n \leftarrow$ the length of $X$
    $M \leftarrow 0$
1:  /* for 2-segment linear regression ($k = 2$) */
2:  for $j = 1, 2, \dots n$ do
3:      $\gamma_{1,j} = \arg\min_{(1<m\leq j)}\{\sigma^2_{1,m} + \sigma^2_{(m+1),j}\}$
4:      $\rho^2_{1,j} = \sigma^2_{1,(\gamma_{1,j}-1)} + \sigma^2_{\gamma_{(1,j)},j}$
5:      $M[1,j] = (\gamma_{1,j}, \rho^2_{1,j})$
6:  end for
7:  /* for $k$-segment linear regression ($k \geq 3$ )*/
8:  for $i = 3, \dots k-1$ do
9:      for $j = 1, 2, \dots n$ do
10:         $\gamma_{i,j} = \arg\min_{(1<m\leq j)}\{\rho^2_{i-1,m} + \sigma^2_{(m+1),j}\}$
11:         $\rho^2_{i,j} = \rho^2_{(i-1),(\gamma_{i,j}-1)} + \sigma^2_{\gamma_{(i,j)},j}$
12:         $M[i,j] = (\gamma_{i,j}, \rho^2_{i,j})$
13:     end for
14: end for
15: /* backtrack $\gamma$ for the pivots from the matrix $M$ */
16: $pivot\_set \leftarrow \{\}$
17: $p\_seg \leftarrow k$
18: $p\_cur \leftarrow n$
19: while $p\_seg > 0$ do
20:     $pivot \leftarrow \gamma_{p\_seg,p\_cur}$
21:     push $pivot$ into $pivot\_set$
22:     $p\_cur \leftarrow pivot - 1$
23:     $p\_seg \leftarrow p\_seg - 1$
24: end while
25: $gse \leftarrow \rho^2_{k,n}$
Output: $pivot\_set$, $gse$
```

the optimal solution in the matrix M for each i-segment is O(kn^2), as shown in Algorithm 2 (Line:8 $\rightarrow$ Line:14). Without doubt, deriving the optimal k-segment partition implies that the time complexity of seeking the pivot point is O(k), as shown in Algorithm 2 (Line:16 $\rightarrow$ Line:24). Finally, taking into account the order of computational complexity of the all operations would be O(n^2) + O(kn^2) + O(k) $\equiv$ O(kn^2). □

V. Experimental results

This section shows the time series datasets considered to be evaluated in the different methods, including OMSLR, Bottom–up (BU) and Brute-force (BF) algorithm. And the experimental settings and the results are also described. These three methods has been implemented by python 3.9 and all the experiments were run using an Intel(R) Core(TM) i7-870K CPU at 3.70 GHz with 128 GB of RAM. These datasets, python code of the algorithms and the experimental results can be accessed publicly at https://github.com/CSCLabTW/TimeSeriesSeg.

A. Datasets used

The performance of OMSLR has been evaluated and compared to other two algorithms whose performance analysis has been evaluated as well. For this purpose, several synthetic and real-world financial time series datasets have been used.

1) synthetic datasets that formed by a random function, range[0, 100], to which random noise (± 5) can be added to produce an infinite number of datasets. In this analysis, we have considered two observation groups, the first set contains 10 series datasets, each has a length of 80 observation points. And another dataset contains a length of 1,000 observations.

2) stock datasets from financial applications, including two different series of S&P 500 index price data; 1. First one is a small size sample that contains 42 data points, and spans a period from 2008-08-01 to 2008-09-30 selected from historical daily price data. 2. Second one is S&P 500 index historical 1-minute price data from 2010-07-01 to 2010-07-07. The dataset contains a length of 4,174 points. Note that the 1-minute price dataset is private, our public datasets do not include this one.

B. Performance measures

1) synthetic datasets for optimality validation: We provide an experimental evaluation of these algorithms, i.e., our OMSLR, BF and the BU algorithm, for examining the performance in terms of running time, Global Square Error (GSE) and the various value of k. GSE is defined in Section III. And BF method is used to give optimal solutions, as ground truth solutions in performance evaluation. To demonstrate the optimality of OMSLR's solution, the comparisons between the ground truth and OMSLR's solution are illustrated.

In the first experiment, we focus on analyzing the optimality of OMSLR's solution and comparing these methods on parameters, $k = 4, n = 80$. Table I shows the results of 10 synthetic series datasets. In Table I, we present simulation evidence to show that the GMS values of the two methods (i.e., BF and our OMSLR algorithm) are optimal, the results are consistent with the Theorem 1 obtained. The performance of BU algorithm is also evaluated. The GSE threshold is setting to the optimal GSE obtained by the BF solution, the number of representation segments by BU algorithm is listed in Table I. It implies that BU algorithm requires more segments than the optimization solution requested to fit the criteria. The GSE granted by BU algorithm is also listed in Table I, all results meet the GSE criteria obtained by the BF solution.

In the second experiment, we show that the performance analysis of varies k value in the large sample size scenario.

TABLE I: The optimality of OMSLR, Bottom-up (BU) and Brute-force (BF) algorithm for 80 data points, $k = 4$

no.	GSE by BF	GSE by OMSLR	k by OMSLR	k by BU[a]	GSE by BU[b]
1	9.4866	9.4866	4	6	8.3408
2	8.7527	8.7527	4	5	8.7527
3	10.4896	10.4896	4	7	10.055
4	6.7802	6.7802	4	9	6.6073
5	7.7435	7.7435	4	8	7.4235
6	7.6787	7.6787	4	9	7.2568
7	6.7738	6.7738	4	8	6.3662
8	8.6074	8.6074	4	6	8.5387
9	9.9907	9.9907	4	7	9.4988
10	7.8571	7.8571	4	6	7.6869

[a] The number of segments is represented by BU algorithm when the fit error threshold is setting to the optimal GSE obtained by the BF solution.

[b] The fit error threshold to BU algorithm is setting to the optimal GSE obtained by the BF solution.

TABLE II: The running time evaluation in 1,000 data points, $k = 1 \rightarrow 10$

k	OMSLR	Brute-force	Bottom-up
1	4.2661s	0.6886s	0.17439s
2	4.4901s	384.2674s	0.13377s
3	4.6303s	145077.2859s[c]	0.13347s
4	4.8120s	N/A	0.13196s
5	4.9905s	N/A	0.13233s
6	5.1748s	N/A	0.13194s
7	5.3544s	N/A	0.13103s
8	5.5345s	N/A	0.13127s
9	5.7158s	N/A	0.13280s
10	5.8947s	N/A	0.13327s

[c] it is more than 40 hours.

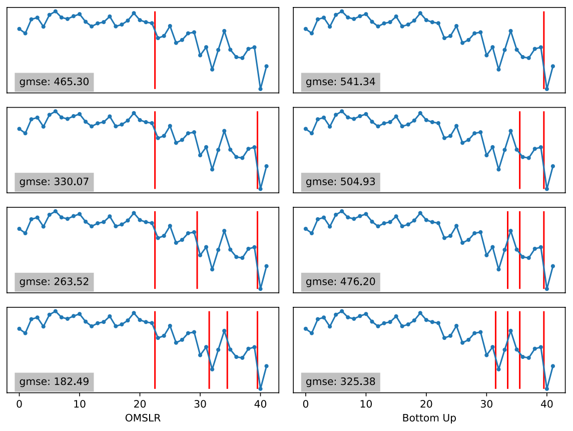
We compare the running time of calculating GSE by OMSLR, BF and BU algorithm for each k from 1 to 10. Table II shows the experimental results, it demonstrates that BU algorithm preforms the better performance than other two algorithms in every experiment. And the running time of OMSLR increased slightly. Compared to other algorithms, BF algorithm needs more than 40 hours to gain an optimal solution when k is only setting to 3. It meets our expectations. When BU algorithm performs the 1-segment representation, the running time of the segment representation is more than others. The reason for the phenomenon is that BU algorithm is a greedy merge based algorithm, more merged operations require less computing time.

2) *financial time series applications:* We provide an experimental evaluation of the two algorithms, i.e., our OMSLR and the BU algorithm [10]. In this experimental results, we examine the performance in terms of Global Mean Square Error (GMSE), with respect to the value of k. The two results are summarized in Fig 1.

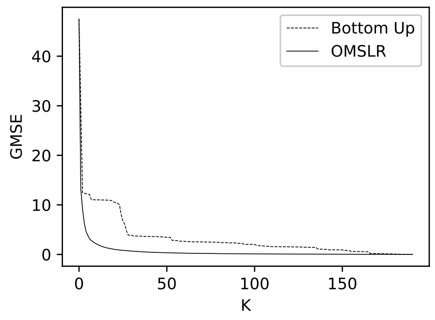
In the first experiment, we compare the step-by-step segment partition as k varies from 2 to 5 in Fig. 1 (a). For illustration purposes, we plot the time series with small sample size. In Fig. 1 (a), it is shown that OMSLR has smaller GMSE than the BU algorithm. The data contains 42 data points, and spans a period from 2008-08-01 to 2008-09-30 selected from the historical daily price data. It also shows that OMSLR always maintains optimality in partitioning the time series into multi-segment linear representation for each value of k.

In the second experiment, we focus on analyzing the relationship between k and GMSE with a large sample size. We use the historical 1-minute price data with a total of 1,560 data points. We compare the GMSE calculated by OMSLR and BU algorithm for each k from 1 to 200. Fig. 1 (b) demonstrates that GMSE generated by the two algorithms both decreases monotonically, and sharply at the beginning.

Therefore, a searching method can be designed for locating the best value of k with a given GMSE bound since the curve is a monotonically decreasing function. Compared to BU algorithm, a much smaller number of segments is required for algorithm OMSLR to find a multi-segment linear regression representation of the given time series to satisfy a given GMSE bound.



(a) The segmentations of OMSLR and BU methods



(b) The comparison of GMSE

Fig. 1: The experimental results.

VI. Conclusion and Future work

In this paper, we proposed a new segmentation algorithm that successfully reduces the computational complexity of the classic optimal method from $O(kn^3)$ to $O(kn^2)$. We study the problem of optimal segmentation of financial time series based on segmented linear regression models. We present the OMSLR algorithm based on the novel two-level DP design. We show that the OMSLR algorithm is optimum with time complexity $O(kn^2)$. We also demonstrate its application in analyzing financial time series. The representation generated by the OMSLR algorithm may be fed into other intelligent applications, e.g., to predict future trend of a financial market. The algorithm may also find further applications. In the future work, we will use it as a benchmark for other stock trading algorithms. And the on-line version of the OMSLR algorithm will be conducted to be used in real-time stock trading. We will also use the OMSLR algorithm in processing data of other time-critical application domains, such as medical and digital health applications.

Acknowledgment

The authors would like to thank Prof. Eamonn Keogh for insightful comments on an earlier draft of this paper.

Appendix A
The derivation of $\alpha_{i,j}, \beta_{i,j}$ and $\sigma^2_{i,j}$.

Note that we have $\alpha_{i,j}, \beta_{i,j}$ and $\sigma^2_{i,j}$ as follows.

$$\begin{aligned}
\alpha_{i,j} &= \bar{x}_{i,j} - \beta_{i,j} \times \bar{t}_{i,j} \\
\beta_{i,j} &= \frac{\sum_{m=i}^{j}(x_m - \bar{x}_{i,j})(m - \bar{t}_{i,j})}{\sum_{m=i}^{j}(m - \bar{t}_{i,j})^2} \\
\sigma^2_{i,j} &= \sum_{m=1}^{j}(x_m - \mu(i,j,m))^2 \\
&= \sum_{m=1}^{j}(x_m - \beta_{i,j} \times m - \alpha_{i,j})^2,
\end{aligned}$$

where $\bar{t}_{i,j} = \frac{(i+j)}{2}$, $\bar{x}_{i,j} = \frac{\sum_{m=i}^{j} x_m}{j-i+1}$, and $\mu(i,j,m) = \beta_{i,j} \times m + \alpha_{i,j}, i \le m \le j$. We can rewrite the above equations into iteration form as follows.

$$\begin{aligned}
\bar{t}_{i,j+1} &= \frac{(i+j+1)}{2} \\
\bar{x}_{i,j+1} &= \frac{\sum_{m=i}^{j+1} x_m}{j-i+2} = \frac{\sum_{m=i}^{j} x_m + x_{j+1}}{j-i+2} \\
&= \frac{\bar{x}_{i,j}(j-i+1) + x_{j+1}}{j-i+2} \\
\alpha_{i,j+1} &= \bar{x}_{i,j+1} - (\beta_{i,j+1} \times \bar{t}_{i,j+1}).
\end{aligned}$$

To compute $\beta_{i,j+1}$, we define $p_{i,j}$ and $q_{i,j}$, thus $\beta_{i,j} = \frac{p_{i,j}}{q_{i,j}}$, where,

$$\begin{aligned}
p_{i,j} &= \sum_{m=i}^{j}(x_m - \bar{x}_{i,j})(m - \bar{t}_{i,j}) \\
q_{i,j} &= \sum_{m=i}^{j+1}(m - \bar{t}_{i,j+1})^2.
\end{aligned}$$

Then, we derive iterative formula of computing $p_{i,j}$ in Equation 9 and $q_{i,j}$ in Equation 10. Note that $\sum_{m=i}^{j+1}(m - \bar{t}_{i,j+1}) = 0$. To compute $\sigma^2_{i,j+1}$ from $\sigma^2_{i,j}$, we define

$$\begin{aligned}
\zeta_{i,j} &= \sum_{m=i}^{j} x_{m^2}, \\
\xi_{i,j} &= \sum_{m=i}^{j} m^2, \\
\delta_{i,j} &= \sum_{m=i}^{j} x_m m, \\
\eta_{i,j} = \sigma^2_{i,j} &= \sum_{m=i}^{j}(x_m - \beta_{i,j} \times m - \alpha_{i,j})^2.
\end{aligned}$$

We then have the following equations,

$$\begin{aligned}
\zeta_{i,j+1} &= \zeta_{i,j} + x_{m^2}, \\
\xi_{i,j+1} &= \xi_{i,j} + m^2, \\
\delta_{i,j+1} &= \delta_{i,j} + x_m m.
\end{aligned}$$

We derive iterative formula of computing $\eta_{i,j+1}$ in Equation 11.

References

[1] J. J. Murphy, "Technical analysis of the financial markets: A comprehensive guide to trading methods and applications." New York Institute of Finance, January 1999.
[2] A. N. Akansu, S. R. Kulkarni, and D. M. Malioutov, "Financial signal processing and machine learning." Wiley-IEEE Press, May 2016.
[3] Y. Abu-Mostafa and A. F. Atiya, "Introduction to financial forecasting," *Applied Intelligence*, vol. 6, pp. 205–213, 2004.
[4] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan, "Mining of concurrent text and time series," in *ACM SIGKDD Workshop on Text Mining*, 2000, pp. 37–44.
[5] C.-H. L. Lee, A. Liu, and W.-S. Chen, "Pattern discovery of fuzzy time series for financial prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 5, pp. 613–625, 2006.
[6] C. Cortes, K. Fisher, D. Pregibon, and A. Rogers, "Hancock: A language for extracting signatures from data streams," in *ACM SIGKDD*, 2000, p. 9–17.
[7] E. Bingham, A. Gionis, N. Haiminen, H. Hiisilä, H. Mannila, and E. Terzi, "Segmentation and dimensionality reduction," in *SIAM SDM*, pp. 372–383.
[8] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surv.*, vol. 45, no. 1, Dec. 2012.
[9] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Commun. ACM*, vol. 4, no. 6, p. 284, Jun. 1961.
[10] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," in *Data Mining in Time Series Databases*, 2004, pp. 1–21.
[11] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in *IEEE ICDE*, 1996, pp. 536–545.
[12] G. Rosman, M. Volkov, D. Feldman, J. W. F. III, and D. Rus, "Coresets for k-segmentation of streaming data," in *NIPS*, Cambridge, MA, USA, 2014, p. 559–567.
[13] J. Acharya, I. Diakonikolas, J. Li, and L. Schmidt, "Fast algorithms for segmented regression," in *ICML*, 2016, p. 2878–2886.
[14] E. J. Keogh and M. J. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *KDD*, 1998, p. 239–243.

$$
\begin{aligned}
p_{i,j+1} &= \sum_{m=i}^{j+1}(x_m - \bar{x}_{i,j+1})(m - \bar{t}_{i,j+1}) \\
&= \sum_{m=i}^{j+1}\Big\{[(x_m - \bar{x}_{i,j}) + (\bar{x}_{i,j} - \bar{x}_{i,j+1})] \times [m - \bar{t}_{i,j} + (\bar{t}_{i,j} - \bar{t}_{i,j+1})]\Big\} \\
&= \sum_{m=i}^{j+1}\Big\{(x_m - \bar{x}_{i,j})(m - \bar{t}_{i,j}) + (\bar{t}_{i,j} - \bar{t}_{i,j+1})\sum_{m=i}^{j+1}(x_m - \bar{x}_{i,j}) + (\bar{x}_{i,j} - \bar{x}_{i,j+1})\sum_{m=i}^{j+1}(x_m - \bar{t}_{i,j}) + (j-i+1)(\bar{x}_{i,j} - \bar{x}_{i,j+1})(\bar{t}_{i,j} - \bar{t}_{i,j+1})\Big\} \\
&= p_{i,j} + (j-i+1)(\bar{x}_{i,j} - \bar{x}_{i,j+1})(\bar{t}_{i,j} - \bar{t}_{i,j+1}) + (x_{j+1} - \bar{x}_{i,j})(j+1-\bar{t}_{i,j})
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
q_{i,j+1} &= \sum_{m=i}^{j+1}(m - \bar{t}_{i,j+1})^2 \\
&= \sum_{m=i}^{j+1}(m - \bar{t}_{i,j} + \bar{t}_{i,j} - \bar{t}_{i,j+1})^2 \\
&= \sum_{m=i}^{j+1}(m - \bar{t}_{i,j})^2 + 2(\bar{t}_{i,j} - \bar{t}_{i,j+1})\sum_{m=i}^{j+1}(m - \bar{t}_{i,j}) + (\bar{t}_{i,j} - \bar{t}_{i,j+1})^2 \\
&= q_{i,j} + (j+1-\bar{t}_{i,j})^2 + 2(\bar{t}_{i,j} - \bar{t}_{i,j+1})\sum_{m=i}^{j+1}(m - \bar{t}_{i,j+1} + \bar{t}_{i,j+1} - \bar{t}_{i,j}) + (\bar{t}_{i,j} - \bar{t}_{i,j+1})^2 \\
&= q_{i,j} + (j+1-\bar{t}_{i,j})^2 - 2(j-1)(\bar{t}_{i,j+1} - \bar{t}_{i,j})^2 + (\bar{t}_{i,j} - \bar{t}_{i,j+1})^2
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
\eta_{i,j+1} &= \sum_{m=i}^{j+1}(x_m - \beta_{i,j+1} \times m - \alpha_{i,j+1})^2 \\
&= \sum_{m=i}^{j+1}(x_m - (\beta_{i,j+1} \times m - \alpha_{i,j+1}))^2 \\
&= \sum_{m=i}^{j+1}x_{m^2} + \sum_{m=i}^{j+1}(\beta_{i,j+1} \times m - \alpha_{i,j+1})^2 - \sum_{m=i}^{j+1}x_m(\beta_{i,j+1}b \times m - \alpha_{i,j+1}) \\
&= \sum_{m=i}^{j+1}x_{m^2} + 2\beta^2_{i,j+1}\sum_{m=i}^{j+1}m^2 + 2\alpha_{i,j+1}\beta_{i,j+1}\sum_{m=i}^{j+1}m + (j+2-i)\alpha^2_{i,j+1} - 2\beta_{i,j+1}\sum_{m=i}^{j+1}x_m m - 2\alpha_{i,j+1}\sum_{m=i}^{j+1}x_m \\
&= \zeta_{i,j+1} + \beta^2_{i,j+1}\xi_{i,j+1} + 2\alpha_{i,j+1}\beta_{i,j+1}\{\frac{(j+1+i)(j+2-i)}{2}\} + (j+2-i)\alpha^2_{i,j+1} - 2\beta_{i,j+1}\delta_{i,j+1} - 2\alpha_{i,j+1}(j+2-i)\bar{x}_{i,j+1}
\end{aligned}
\tag{11}
$$

[15] I. DOMOWITZ, J. GLEN, and A. MADHAVAN, "Market segmentation and stock prices: Evidence from an emerging market," *The Journal of Finance*, vol. 52, no. 3, pp. 1059–1085, 1997.

[16] E. Meijering, "A chronology of interpolation: from ancient astronomy to modern signal and image processing," *Proceedings of the IEEE*, vol. 90, no. 3, pp. 319–342, 2002.

[17] L. Martí, N. Sanchez-Pi, J. M. Molina, and A. C. Bicharra Garcia, "Yasa: Yet another time series segmentation algorithm for anomaly detection in big data problems," in *Hybrid Artificial Intelligence Systems*, M. Polycarpou, A. C. P. L. F. de Carvalho, J.-S. Pan, M. Woźniak, H. Quintian, and E. Corchado, Eds. Cham: Springer International Publishing, 2014, pp. 697–708.

[18] H. Aksoy, A. Gedikli, N. E. Unal, and A. Kehagias, "Fast segmentation algorithms for long hydrometeorological time series," *Hydrological Processes*, vol. 22, no. 23, pp. 4600–4608, 2008.

[19] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, and Y. Mu, "Piecewise linear approximation of streaming time series data with max-error guarantees," in *IEEE ICDE*, 2015, pp. 173–184.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009.

lines," *Journal of the American Statistical Association*, vol. 64,

[21] R. Bellman and R. Roth, "Curve fitting by segmented straight

no. 327, pp. 1079–1084, 1969.

[22] E. Terzi and P. Tsaparas, *Efficient algorithms for sequence segmentation*, pp. 316–327.

[23] S. Guha, N. Koudas, and K. Shim, "Data-streams and histograms," in *ACM STOC*, 2001, p. 471–475.

[24] Ángel Carmona-Poyato, N. L. Fernández-Garcia, F. J. Madrid-Cuevas, and A. M. Durán-Rosal, "A new approach for optimal offline time-series segmentation with error bound guarantee," *Pattern Recognition*, vol. 115, p. 107917, 2021.

[25] F. lai Chung, T. chung Fu, R. Luk, and V. Ng, "Evolutionary time series segmentation for stock data mining," in *IEEE ICDM*, 2002, pp. 83–90.

[26] J. Yin, Y.-W. Si, and Z. Gong, "Financial time series segmentation based on turning points," in *ICSSE*, 2011, pp. 394–399.

[27] C. Dose and S. Cincotti, "Clustering of financial time series with application to index and enhanced index tracking portfolio," *Physica A: Statistical Mechanics and its Applications*, vol. 355, no. 1, pp. 145–151, 2005, market Dynamics and Quantitative Economics.

[28] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.