Structured Time Series Stream Data

Ph.D. Thesis Proposal

Huanmei Wu
maggiewu@ccs.neu.edu
College of Computer and Information Science
Northeastern University
Boston, MA 02115

Thesis Advisor: Prof. Betty Salzberg

Abstract

Management of time series stream data is an important technique for database, networks, operating systems, and theoretical research. Analysis of time series stream data is widely used for many applications such as economic forecasting, stock market analysis, budgetary analysis, and workload projects. Subsequence matching in large time series database tries to find subsequences similar to a query sequence, which is a technique widely used in pattern matching, motion prediction, and rule discovery. Both management and subsequence matching are difficult to achieve because the constant changes, sheer volume, and complex data structures of massive stream data. Our objectives include: (i) modeling and analyzing time series stream data with consideration of its internal structure, (ii) developing efficient and accurate algorithms for piecewise linear representations of data streams in an online fashion, (iii) using internal structure of a time series directly in online subsequence similarity matching over data streams, and (iv) applying statistical methods for retrieved subsequences analysis.

We have developed a framework to achieve these objectives. The framework is composed of (a) a piecewise linear model which captures the internal structure of a time series stream data, (b) an online segmentation and pruning algorithm which produces the piecewise linear representation of raw stream data in real-time, (c) an online subsequence matching engine which generates dynamic query subsequences, defines application specific subsequence similarity measures and performs similarity matching with consideration of the internal structure of the time series and (d) statistical and probabilistic techniques which analyze similarity matching results for many specific applications, such as motion characterization, future prediction, correlation discovery and clustering.

This framework covers all application domains with structured time series stream data. The techniques can be used for real-time systems where response time is critical. We have applied the framework to multiple problem domains, such as financial data analysis and tumor respiratory motion characterization (used in cancer radiation treatment).

1 Introduction

Time series stream data often arise when following industrial processes, monitoring patient treatments, or tracking corporate business metrics. Analysis of time series stream data is widely used for many applications such as economic forecasting, stock market analysis, process and quality control, budgetary analysis, and workload projections. This thesis presents my study on structured time series stream data, including...
modeling, segmenting, subsequence similarity matching, and analyzing a time series in an online fashion, with adequate consideration the internal structure of the time series.

1.1 Motivation

Time series stream databases are distinguished from conventional databases in several aspects. First, stream data are continuously generated/modified. Raw data is too large to be stored in a traditional database for efficient data management. Second, querying on stream databases is difficult in set-oriented data management systems. Third, a single-pass search over the stream is mandatory since it is infeasible or impossible to rewind the stream. Fourth, the answers of a query usually are approximate and partial answers.

In database research, there has been an explosion of interest on time series databases. Many high level representations of time series [4, 5, 6, 8, 12, 17, 22, 30], and distance functions for subsequence matching [2, 6, 24, 28] have been proposed. However, far less attention has been paid to the internal structure within the data. The internal structure of a time series affects decision-making and influences data processing. Here we present two motivating examples to manifest the necessity to consider the internal structure of the time series.

**Example 1:** Figure 2 illustrates gated treatment in image guided cancer treatment based on tumor respiratory motion. The gating window does not move. The tumor moves in and out of the gating window. The treatment is on when the tumor is in the gating window. It is more important to characterize motion for the portion in the gating window than other portion of the tumor motion. The internal structure of tumor motion will help in designing better treatment planning and providing better patient care.

**Example 2:** Figure 1 shows the piecewise linear representation of some stock data. The two subsequences would be considered similar using existing subsequence similarity measures. But technical analysis of financial data is also concerned with the relative position of the upper end points as well as the relative position of the lower end points. The two subsequences in Figure 1 would not be considered similar by financial data analysts because point 1 and point 3 are not in the same relative positions as point 1’ and point 3’. This example shows the new requirement for consideration of the internal structure during subsequence similarity matching for specific applications.

1.2 General Problems

Research on time series stream data is a broad topic. This section gives a brief summary of the current status of research on modeling, segmenting, matching and clustering time series data.

Internal structures of a time series can be modeled into two categories: periodic and non-periodic structures. Many time series display periodic fluctuations. For example, temperature of a region tends to peak in

![Figure 1: Subsequence similarity with different relative positions.](image1.png)

![Figure 2: Gated treatment in tumor radiation treatment.](image2.png)
the summer and then declines in the fall. It reaches the lowest in the winter and then climbs up in the spring. So time series of temperature of a region will typically show the periodical seasonal changes. Periodicity is also quite common in economic time series and some medical time series. In addition to periodicity, there are other non-periodic structures in time series, such as trends, correlation and autocorrelation among time series data streams.

**Piecewise linear representation** (PLR) uses line segments to approximate time series [7, 14]. So a time series $T$ of length $n$ will be approximated with $k$ straight line segments, where $k \ll n$. This representation makes the storage much smaller and computation more efficient. PLR provides a convenient tool for effective solutions, since it supports fast similarity matching, new definitions of distance measures of time series, and novel clustering and classification algorithms.

**Subsequence matching** methods try to find subsequences similar to a query sequence within a large time series databases. They are important in data mining and are used for pattern matching, future movement prediction, new pattern identification, rule discovery and computer aided diagnosis. Subsequence matching has attracted recent interest and several solutions have been proposed [2, 4, 6, 13, 15, 19, 24, 28]. Research in time series *stream data* is still in its preliminary stage. Stream data are naturally ordered in time. Some streams are ordered in a fixed time interval and can be treated as stream time series directly. Some streams come in irregularly and special procedures are needed in order to apply time series techniques. For example, there are thousands of stock transactions every second, which may be carried out at any time and there are different numbers of transactions at different times.

**Clustering** is a process to group a set of objects into a set of meaningful subclasses (called clusters). Data objects are similar to one another within the same cluster and dissimilar to the objects in other clusters. Cluster analysis is one of the major topics in data mining. General applications of clustering include statistics, pattern recognition, spatial data analysis, image processing, market research, and WWW document classification. For instance, market analysts discover distinct groups in their customer databases and characterize customer groups based on purchasing patterns. It is very difficult to do cluster analysis in data streams because of its large data volume. Several one-pass stream clustering algorithms have been developed recently [9, 21]. Based upon the previous work, a new clustering process with the combination of an online micro-clustering and an offline clustering components is proposed [1].

### 1.3 Challenges

Online processing of structured time series stream data requires special concerns in modeling the internal structure, segmenting the data online, and considering internal structure directly during similarity matching. This section introduces the challenges for structured time series stream data processing.

**Internal structure Modeling**

Internal structure gives meaning to a time series, which allows for a more accurate forecast. However, structured data generally contains random noise which makes it difficult to analyze a time series in an online fashion. Randomness cannot be predicted and it limits the certainty of future prediction.

Previous research on time series data has not explicitly considered the meaning of the time series data and its influence over similarity matching. The internal structure and its meaning should be modeled and analyzed for two reasons. First, a model can be used in forecasting and monitoring a time series. Second, it helps to obtain an understanding of the underlying forces and structure that produced the corresponding time series.

Modeling a time series is a very challenging work. First the model must be a good representation of its internal structure and be able to preserve the natural meaning of the internal structure. This needs thorough understanding of the underlying applications. Second, the resulting model should be a compact representa-
tion of raw data so that it will reduce storage for stream data. Third, the model must be able to be generated in an online fashion so that the internal structures can be decided in real-time. This is a general requirement for real-time analysis of time series stream data.

**Piecewise Linear Representation over Stream Data**
Segmentation over data streams has its special requirements. First, due to massive incoming data, a one-pass algorithm is mandatory since it is infeasible to roll back to all the raw historical data. In this one pass, raw data is separated into a manageable number of line segments. The produced PLR segments can be accessed multiple times for later sequence comparison.

Second, the algorithm must work in an online fashion. Offline segmentation algorithms are not applicable for data streams, since new data is coming consecutively. Third, the algorithm should be sensitive to crucial points. At the same time, it should efficiently remove noise signals. Finally, the produced line segments should be in a specified shape, such as the zigzag shape in financial data.

**Online Subsequence Similarity Matching**
Subsequence similarity matching in data streams is more general and difficult. Existing techniques on time series subsequence matching mainly focus on discovering the similarity between an online querying subsequence and a traditional database. The queried data are static and built with an index. Although there is recent research [31, 32] on pair-wise correlated statistics with an online fashion, it is focused on similarity for whole data streams, not subsequence similarity. The base of our subsequence similarity matching of data streams is an up-to-date stream database and online query subsequences. The database will be automatically updated as new stream data comes in. So the database reflects the most recent changes. The query subsequence is automatically generated based on the current state of the data stream. New similarity measures are defined to satisfy the new application specific requirements.

**Dynamic Clustering**
To improve accuracy for subsequence similarity matching, matching over irrelevant or dissimilar streams should be avoided. Also, subsequence similarity matching over all the streams in a massive stream environment is impractical or unrealistic for some real time applications where fast searching and quick response are required. So we need to cluster data streams based on their similarity. Dynamic clustering is necessary because new streams may evolve and old streams may change their patterns.

Clustering data streams is a difficult problem for the data stream domain because of the volume of data in the streams. A clustering algorithm for data streams must satisfy the following conditions. First, data representation should be very compact. Second, processing new data points should be fast and incremental. Third, outliers should be identified and discarded. Fourth, evolving streams should be re-clustered to satisfy new patterns. And at last, for our application, the clustering algorithm should be based on the subsequence similarity matching results, not the whole streams. To the best to our knowledge, no existing clustering algorithm can satisfy all the above requirements. For example, the BIRCH [26] algorithm is not suitable for stream clustering because of its loose data representation. STREAM [9] is not applicable if the data evolves considerably over time. Dynamic stream clustering based on subsequence similarity matching results for recent stream data will be a great benefit in trend prediction, computer aided diagnosis, and environmental data analysis, the applications we address.

**1.4 Proposed Research Contributions**
Our work differs from previous research in several aspects. We address the application-guided research on structured time series stream data. Our proposed framework is general for application domains with
structured time series stream data. Our technologies can be used for real-time systems where response time is critical. We have applied the framework to several application domains, such as stock data analysis and tumor respiratory motion analysis (used in cancer radiation treatment).

2 Related Work

In this section, we will describe, in detail, the major approaches for piecewise linear representation of time series, subsequence similarity measures, and clustering algorithms.

2.1 Piecewise Linear Representation

Generally, the existing segmentation algorithms can be classified into four categories [14]: sliding window algorithms, top-down algorithms, bottom-up algorithms, and sliding window and bottom-up (SWAB) algorithms.

The sliding window algorithm generates a line segment in an online fashion. It fixes the left end point of a potential line segment at the first data point of a time series. Then the line segment will grow until it exceeds some error bound. The subsequence from the left end point to the previous data point will be transformed into a segment. The new left end point will be the newly arrived data point, and the above process repeats until the entire time series has been transformed into a piecewise linear approximation.

The sliding window algorithm is attractive because it is simple and it works in an online fashion. The variations and optimizations of the basic algorithm are particularly popular in medical data analysis, since patient information is inherently an online task.

The top-down algorithm partitions the time series recursively until some stopping criteria is met. In each loop, every possible portioning of the time series is considered and the best splitting is adopted. After a segmentation is made, both subsections are then tested to see if their approximation satisfies some user specified conditions. If not, the algorithm will continue to split the subsections until all the segments have satisfied the stop conditions.

The top-down algorithm works in an offline fashion and needs multiple scan of the time series. Variations on the top-down algorithm have been developed in different research fields. For example, in Park et al. [22], a first scan of the entire dataset only marks every peak and valley, which create the initial segmentation. Then the top-down algorithm is applied to each of the segments. Fink and Pratt [23] used the top-down algorithm to identify major extrema in stock market prediction.

The bottom-up algorithm is complementary to the top-down algorithm. It starts from the finest possible approximation, so that a \( n \) length time series is approximated by \( n/2 \) segments. Then segments are merged iteratively until some stopping criteria are met. Variations on the bottom-up algorithm are very common in the fields of data mining and medicine.

The sliding window and bottom-up algorithm (SWAB) [14] is the combination of the sliding window algorithm and the bottom-up algorithm. Experimental results showed that the sliding window algorithm generally has poor performance and bottom-up often significantly outperforms top-down. This is because the sliding windows algorithm lacks the global view of its offline counterparts. The bottom-up and top-down methods have better results, but they are offline and require multiple scans of the entire data sets. The SWAB algorithm captures the online nature of sliding windows and yet retains the superiority of bottom-up.
The SWAB algorithm keeps a small buffer. Bottom-up is applied to the data in the buffer and the leftmost segment is reported. The data corresponding to the reported segment is removed from the buffer and more data points are read in. These points are incorporated into the buffer and bottom-up is applied again.

2.2 Subsequence Similarity Matching

Similarity search in time series is useful for many data mining applications. Agrawal et al. [2] studied whole sequence similarity matching. Faloutsos et al. [6] generalized it to subsequence similarity matching. The basic idea is to transform the sequence into the frequency domain using a Discrete Fourier Transformation (DFT). Then the first few features are extracted and the Euclidean distance is used as the similarity distance function. Multidimensional indexing methods such as the R*-tree [3] can be applied for fast search. New research based on subsequence search has advanced in several directions. New methods in constructing MBRs reduce false negatives [20]. Other feature extraction functions, such as the Discrete Wavelet Transformation (DWT) [4, 8], and Adaptive Piecewise Constant Approximation (APAC) [12] have been proposed to reduce the dimensionality of time series data. New distance functions such as Dynamic Time Warping [11, 22, 30] have been explored to overcome the brittleness of Euclidean distance measure or its variations [2, 6, 24]. These advances are discussed here:

The Euclidean Similarity Measure views each sequence as a point in n-dimensional Euclidean space. It defines the similarity between two sequences \( X \) and \( Y \) as the \( L_p \) norm:

\[
L_p(X, Y)
\]

The Euclidean model is very easy to compute and it is easy to index and cluster data sets based on Euclidean distances. But it does not allow for different baselines and does not allow for different scales.

The Discrete Fourier Transform (DFT) transforms the data from the time domain to the frequency domain. It highlights the periodicities. It first decomposes signals to a sum of sine (and cosine) waves. According to Parseval’s theorem, DFT preserves the energy, i.e., the Euclidean distance between the original time series and transformed time series are the same. So we can apply DFT to each time series, keeping the first 2-10 coefficients as a feature vector and using an R-tree to index the vectors. The raw time series is approximated by a few frequencies. The resulting data is highly compressed and it can be indexed and used to discover pattern and forecast trends based on the low dimensional frequency features.

The Discrete Wavelet Transform (DWT), similar to the DFT, is based on a decomposition of a signal using an orthonormal family of basis functions. Unlike the DFT, a wavelet has its energy concentrated in time. The DFT is useful in analyzing periodic and time-invariant phenomena, while the DWT is well suited for the analysis of transient, time-varying signals. The advantages of Wavelets is that it has better data compression, is good for progressive transmission, is fast to compute (O(n)!), and it handles spikes well. The DWT feature space keeps the \( d \) most important wavelet coefficients. The feature space is normalized and the largest coefficient is kept.

Dynamic Time Warping (DTW) is extensively used in speech recognition. It allows acceleration and deceleration of signals along the time dimension. For two given time series \( X = \{X_1, X_2, ..., X_n\} \), and \( Y = \{Y_1, Y_2, ..., Y_n\} \). DTW allows each sequence to be extended by repeating elements. Then Euclidean distance is calculated between the extended sequences \( X' \) and \( Y' \). The restriction is that DTW requires monotonicity and continuity. Monotonicity means that a path should not go down or to the left. Continuity means no elements may be skipped in a sequence. DTW also has a restriction on its window size, as shown
Longest Comment Subsequence (LCS) is a technique to search for similarities across multiple texts. It takes a set of strings as an input and outputs the longest common substring. One major application is in finding a consensus among DNA sequences. Dynamic programming can be used to find the longest common subsequence of two strings, S and T, of n and m characters each. Let lcs[i, j] denote the number of characters in the longest common subsequences of S[1], ..., S[i] and T[1], ..., T[j]. Then dynamic programming will calculate lcs[i, j] as the following:

\[
\text{lcs}(i, j) = \begin{cases} 
\max( \text{lcs}(i - 1, j), \text{lcs}(i, j - 1) ) & S[i] \neq T[j] \\
\max( \text{lcs}(i - 1, j - 1) + 1, \text{lcs}(i - 1, j), \text{lcs}(i, j - 1) ) & S[i] = T[j]
\end{cases}
\]

### 2.3 Clustering

Different clustering approaches can be summarized in Figure 4 based on a clustering review in Jain et al. [10]. Clustering approaches can be broadly classified into Hierarchical and partitional approaches. In this section, we will first give a brief introduction of hierarchical and partitional clustering. Then we go into detail on recently developed clustering algorithms for databases.

**Hierarchical Clustering Algorithms**

*Hierarchical clustering algorithms* create a sequence of partitions in which each partition is nested into the next partition in the sequence. In *agglomerative clustering*, two smaller clusters will be merged to form a larger cluster based on some minimum distance criteria. At the beginning, each object is treated as a cluster. Then the algorithms will recursively merge the most similar pair of clusters based on a proximity matrix. The algorithm stops until k clusters is left (k is a predefined parameter).

Most hierarchical Clustering Algorithms are variants of single-link [25] or complete-link [16] algorithms. These two algorithms are different in the way they characterize the similarity between a pair of clusters. For the single-link algorithm, the distance between two clusters is measured as the similarity between the most similar pair of elements, i.e., the *minimum* of the distances between all pairs of patterns drawn from the two clusters. For the complete-link algorithm, the distance between two clusters is measured as the least similar pair of elements, i.e., the *maximum* of the pair-wise distances. The complete-link algorithm produces tightly bound or compact clusters, while the single-link algorithm tends to produce elongated clusters, producing what is called “the chaining effect”.

**Partitional Clustering Algorithms**
Partitional clustering algorithms obtain a single partition of the data by optimizing a criterion function defined either locally or globally. The most intuitive and frequently used criterion function is the squared error criterion, which tends to work well with isolated and compact clusters. The most commonly used algorithms employing a squared error criterion are k-Means [18] and its variants. Their advantages are the quick running time and their robustness to outliers. One major shortcoming of them is they tend to produce spherically shaped clusters of similar size, which prevents the finding of natural clusters.

Sample Clustering Algorithms
This section presents several widely used clustering algorithms including BIRCH [26], STREAM [9], and CluStream [1].

**BIRCH** stands Balanced Iterative Reducing and Clustering using Hierarchies. It is an efficient data clustering method for very large databases. BIRCH uses local decisions to maximize the use of memory and minimize the amount of I/O required. It incrementally and dynamically clusters incoming multi-dimensional metric data points. BIRCH scans the data almost once and is independent of data ordering. It handles outliers gracefully. But BIRCH takes a considerable amount of time, and data representation is not as compact as is required to process data streams.

**STREAM** aims to provide guaranteed performance of data stream clustering. The solution of STREAM is within a factor of \( c \) of the optimal squared distances. It processes data streams in batches of points. The algorithm clusters the points in one batch first and retains the weighted center. Then it clusters all weighted centers to obtain a clustering model. The data representation is compact, but it increases as more batches are processed. The algorithm takes longer than k-Means to find a bounded solution. It can not successfully discover the outliers, which may result in distortion of the real clusters and will effectively mask the emerging trend of clusters.

**CluStream** is developed for clustering large evolving data streams. It is a new framework for clustering evolving data streams with an online component and an offline component. By using the concepts of a pyramidal time frame in conjunction with a micro-clustering approach, the CluStream model provides a wide variety of functionality in characterizing data stream clusters over different time horizons in an evolving environment. The use of a pyramidal time window assures that the essential statistics of evolving data streams can be captured without sacrificing the underlying space- and time-efficiency of the stream clustering process. The exploitation of microclustering ensures that CluStream can achieve higher accuracy than STREAM due to its registering of more detailed information than the \( k \) points used by the k-means approach.

3 Proposed Research
We have proposed a general framework for structured time series stream data for online application and real-time systems. The framework covers modeling, online segmenting, online similarity matching, and analyzing stream data. This section will present the framework first. Then We will customize the framework for real applications, including stock data management and tumor respiratory motion data management (used in cancer radiation treatment).
3.1 Generalized Framework

Our framework for structured time series stream data analysis has considered the internal structure in our data management and data analysis. It has also considered the special properties for online processing of stream data.

**Definition 1 (Framework)** A framework for structured time series stream data has four components:

1. A piecewise linear model (PLM) = \( \{Q, \Sigma, D\} \), where \( Q \) is a finite set of linear states (the number of states in \( Q \) is \( N_Q \)). \( \Sigma \) is a finite set of input alphabet. \( D \) is the transition relationship from one state to another.

2. An online segmentation and pruning algorithm (SPA) = \( \{F_i, F_o, F_p\} \), where \( F_i \) is the function to discover the initial state. \( F_o \) is the segmentation function for transition from one state to another. \( F_p \) is the pruning function for optimizing previous segmentation.

3. An event-driven online subsequence matching engine = \( \{\Omega, F_e\} \), where \( \Omega \) is subsequence similarity measure between two subsequences. \( F_e \) is the function to detect critical events for event-driven similarity matching.

4. A set of application specific functions \( \{F_a\} \) for statistical result analysis.

The four steps are independent from each other and can be modified separately. This section will present the details of each step and the general.

3.1.1 Piecewise Linear Modeling

The piecewise linear model (PLM) must be built for each specific application, since the states of one system are different from those of another. So to build the piecewise linear model, one must investigate thoroughly the underlying application, characterize the produced time series, and probe the internal structures of the time series. These studies will formalize \( Q, \Sigma \) and \( D \) of the framework in Definition 1, i.e., PLM = \( \{Q, \Sigma, D\} \).

For each time series stream data, it is possible there is irregular motion of the data. So for each PLM, there is an irregular state. Data can enter the irregular state and go back to normal states at any time. The transition relationship \( D \) can be demonstrated with a transition diagram. The transition diagram will decide what is the current state for each arriving data point non-deterministically. It is very similar to a non-deterministic finite automaton (NFA). But a PLM transition diagram is distinguished from an NFA by the following two features:

- The initial state of an NFA is a single predetermined state. The initial state of a PLM diagram is one of a set of states, and it cannot be determined in advance. Each time series has to determine its initial state by computing the initial state function \( F_I \) in an online fashion.

- An NFA has a subset of state as the acceptable states. The PLM terminates whatever the input time series terminates. It will accept a time series no matter what the final state is.

All the consecutive data points, which are determined to be in the same state, will be placed into a single line segment. The original state will be the state of the line segment. The state sequence of a time series are the states of all the line segments in order. That is to say, each line segment corresponds to one range of one state. The final PLM representation is a sequence of line segments.

This piecewise linear model captures the internal structure, reduces data storage, and provides a convenient tool for data analysis, and is suitable for application specific similarity measurements and motion prediction. The internal structure defines the states and transition relationships.
3.1.2 Online segmentation and pruning

For stream data, the piecewise linear representation (PLR) must be processed in an online fashion. In addition, the PLR segments should be in a predefined pattern according to the transition diagram $D$. We have developed a new online segmentation and pruning algorithm to produce piecewise linear representation of raw time series stream data. Our new online algorithm takes a raw data stream as input and outputs another stream of PLR line segments. This algorithm is distinguished from others in two ways. First, it adopts an optimization with simultaneous segmentation and pruning. Second the resulting PLR segments satisfy the piecewise linear model with a predefined pattern.

The algorithm is illustrated in Figure 5. Two varied size sliding windows are used for online segmentation and pruning separately. There are predefined criteria to start a new line segment and criteria to prune existing line segments.

The segmentation window covers the data points of the most recent line segment. When there is a state transition from one state to another, a new line segment is generated. In our algorithm, state transitions are primarily triggered by motion velocity changes. Each state has an expected range of velocities, although the ranges for the different states overlap. Besides the velocity changes, there are other associated rules to decide if a transition has occurred or not.

In our online algorithm, segmentation proceeds in a greedy manner. Whenever the segmentation criteria are satisfied, the segmentation will generate a new line segment and transit to a new state. One big advantage of this greedy algorithm is the quick identification of a new state transition point. One disadvantage is that it may generate noisy line segments. The pruning process is to make sure such line segments will be eliminated from the final PLR.

The pruning window covers more data points than the segmentation window does. The pruning window covers the most recent three line segments. Any historical line segments before the pruning window will not be affected by successive procedures. When a new line segment is generated by the segmentation criteria, the pruning process is activated to make sure that the line segments in the pruning window are valid line segments. The pruning criteria and corresponding pruning actions are defined by specific applications. One general requirement for the pruning process is to keep the original pattern of the internal structure.

This online segmentation and pruning algorithm identifies critical events efficiently, and produces piecewise linear presentation accurately. In addition, it is a one-pass algorithm, which runs in linear time and with constant space. Thus it is suitable for both online and offline data analysis. These tools can be effectively used in online decision-making and real-time applications.

3.1.3 Online subsequence matching

There are special requirements for online subsequence similarity matching over structured time series stream data generated by different applications. The subsequence similarity measure should consider these special requirements explicitly. There are also concerns about how to generate query subsequences dynamically.
and when to perform subsequence similarity. In this section, we will discuss subsequence similarity first. After that we present our ideas for dynamic query subsequence generation. Then we introduce a new online event-drive matching mechanism.

**Subsequence Similarity Measure**

Subsequence matching is based on the piecewise linear representation (PLR) of raw data. Each vertex \( v_i \) is represented by three elements: \((t_i, x_i, s_i)\). The vertex time \( t_i \) both denotes the start time of a line segment (beginning with the vertex) and the end time of the previous line segment (terminated with the vertex). The space position of \( v_i \) is denoted \( x_i \). Since data point have different dimensionalities (i.e., 1-D for stock market, 1-D, 2-D and 3-D for tumor respiratory motion), we have proposed an approach that can work for any \( n \)-dimensional space. Thus \( x_i \) can be an \( n \) dimensional point. Last, \( s_i \) is the state of the line segment beginning with vertex \( v_i \). Based on this data model, we have proposed a subsequence similarity measure as follows.

**Definition 2 (Subsequence Similarity \( \Omega \))** Two subsequences with the same length \( n \):

\[
S_1 = [(t_{11}, x_{11}, s_{11}), \ldots, (t_{1n}, x_{1n}, s_{1n})] \quad \text{and} \quad S_2 = [(t_{21}, x_{21}, s_{21}), \ldots, (t_{2n}, x_{2n}, s_{2n})]
\]

are similar if they satisfy the following conditions:

1. \( f(S_1, S_2) = \text{true} \), where \( f(S_1, S_2) \) determines if the two subsequences have the same internal structure and \( f(S_1, S_2) \) is dependent on the underlying application.

2. \( d_s(S_1, S_2) < \delta \), where \( d_s(S_1, S_2) \) is the weighted online subsequence distance function defined as:

\[
d_s(S_1, S_2) = \frac{1}{(n - 1)w_s} \left[ \alpha \sum_{i=1}^{n-1} (w_i \cdot \Delta x_i) + \beta \sum_{i=1}^{n-1} (w_i \cdot \Delta t_i) \right], \text{ where}
\]

\[
\Delta x_i = | |x_{1(i+1)} - x_{1i}| - |x_{2(i+1)} - x_{2i}| |, \quad \Delta t_i = | (t_{1(i+1)} - t_{1i}) - (t_{2(i+1)} - t_{2i}) |
\]

and \( \alpha \) and \( \beta \) are the different weights for the amplitude and frequency, \( w_i \) is the weight for vertex \( v_i \), and \( w_s \) is the weight between the two subsequences.

This model-based, multi-layer, weighted, and parametric similarity definition has addressed the general concerns for structured time series stream data. Condition 1 ensures that the corresponding subsequences have the same internal structure. That is one big advantage of using the piecewise linear representation based on the piecewise linear model. In condition 2, \( w_s \) gives different weights for pair of subsequences with different relationships. Different values of \( \alpha \) and \( \beta \) provide a tradeoff in the relative importance of amplitude and frequency. The weight \( w_i \) assigns different levels of importance to different portions of the compared subsequences. The more recent fragments more closely represent the current pattern, and have more influence on future move prediction with a larger \( w_i \).

Another salient feature of our similarity measure is its flexibility. It can be applied to different application domains by adjusting the parameters of \( w_s, w_i, \alpha \) and \( \beta \). For instance, the weighted distance functions in [15, 28] are special cases of our distance measure.

**Dynamic query subsequence generation**

But how do we determine the appropriate length of the query subsequence? Shorter query subsequences lower the quality of the representation of the current motion characteristics. Longer query subsequences require additional computation and introduce longer delay. There is a tradeoff between response time and
the length of a query subsequence. For repetitive patterns with periodic structures, we proposed a new concept to produce more representative query subsequences with dynamically adjusted lengths. A flexible scheme is applied to adjust the query subsequence length dynamically based on *subsequence stability*, which is defined as:

**Definition 3 (Subsequence Stability)** Given a subsequence \( S = [(t_1, x_1, s_1), \ldots, (t_n, x_n, s_n)] \), \( S \) is stable if \( \rho(S) < \rho_c \), where \( \rho_c \) is a predefined parameter and \( \rho(S) \) is computed by the following formula:

\[
\rho(S) = \frac{1}{n} \sum_{k=0}^{N_Q} \sum_{i=1, s_i = k}^{n} \left( \alpha \cdot |x_{i+1} - x_i| - \Delta \tau_k \right) + \beta \cdot \left( t_{i+1} - t_i - \Delta t_k \right)
\]

where \( k = 0, \ldots, N_Q \) for each state in \( Q \) of the PLM, and the inner sum is computed over line segments from vertex \( v_i \) to \( v_{i+1} \), where \( s_i = k \) as indicated. \( \Delta \tau_k \) is the average time interval in \( S \) for state \( k \), and \( \Delta \tau_k \) is the average amplitude of \( S \) for state \( k \). \( \alpha \) and \( \beta \) are different weights for amplitude and frequency changes. \( \rho(S) \) is called the stability of \( S \). The smaller \( \rho(S) \) is, the more stability \( S \) has.

The more stable the most recent subsequence is, the shorter the query subsequence will be. The length of the query subsequence is between the user specified minimum length \( L_{min} \) and the maximum length \( L_{max} \). For example, in Figure 6a, \( L_{min} = 3 \) and \( L_{max} = 8 \) repetitive cycles, for a repetitive pattern which has three states. We use a stability checking strip to determine the length of the query subsequence. A stability checking strip is a window of fixed size \( L_{min} \), moving from the most recent portion back to historical data. The subsequence stability of the strip is checked after each move to determine the start point of a query subsequence. As illustrated in Figure 6, in the beginning, the strip covers the subsequence of the most recent \( L_{min} \) vertices. The stability of that subsequence is checked. If the subsequence is stable, the strip halts. If not, the strip will move one cycle back to history data, and stability on the new strip is checked again. This will go on until one of the two conditions is met: a stable subsequence is found (Figure 6a), or there are \( L_{max} \) vertices for the query subsequence (Figure 6b). The query subsequence is from the beginning vertex of the last strip to the most recent vertex. As a consequence, breathing with high regularity will have shorter query sequences, while breathing with low regularity tends to have longer query subsequences.

**Event-driven subsequence matching**

Time series stream data comes in continuously. Performing similarity search upon all incoming data is not efficient for massive stream data management, especially not for real-application such as stock market analysis. A naive solution is to search after a fixed time period. This will reduce the computational burden but it is insensitive to the changes between adjacent query times and may lose some potentially important information. We propose an event-driven online subsequence similarity matching mechanism for time-sensitive applications.

In event-driven online subsequence matching, the search engine is triggered only if there is a critical event. A critical **event** is defined by specific applications. For each application, there is an alerting function \( F_n \) which can detect a potential critical event. Once the conditions are satisfied, the search engine is activated.

**3.1.4 Statistical analysis**

Analyzing the retrieved similar subsequences is significant for specific applications. We apply different statistical and probability techniques for result analysis, which can be used for many applications. Sample applications includes but are not limited to: motion characterization (such as detection of new patterns, assessment of the reliability of an old pattern, and quantification of amplitude and frequency variations),
motion prediction (such as trend prediction of stock market, gating signal on/off prediction of cancer treatment, and exact tumor position prediction), correlation discovery (such as trend correlation between two stocks and physiological correlation between respiratory motion and other patient information) and clustering (such as grouping financial streams with similar trends and patients with similar respiratory motion patterns).

3.2 Stock market data analysis

This section shows how to use our framework for stock market data analysis. For financial data [28], a normal stream moves with two regular states: UP and DOWN. UP corresponds to a bull market, while DOWN corresponds to a bear market. The irregular state happens when the trend of the market is unpredictable, called NOTREND state. Thus in the PLM of the stock market analysis, \( Q = \{ \text{UP, DOWN, NOTREND} \} \), \( \Sigma \) is a finite set of positive real numbers, and the transition diagram \( D \) is illustrated in Figure 7a.

For a regular stock market, the resulting PLR segments should be in an up-down-up-down repetitive pattern (the zigzag shape). An upper point is defined as follows (the definition of a lower point is symmetric) and is shown in Figure 8.

Suppose the current point is \( P_j(X_j,t_j) \). The upper point \( P_i(X_i,t_i) \) is a point in the current sliding window that satisfies: (1) \( X_i = \max \text{ X values of current sliding window } \); (2) \( X_i > X_j + \delta \) (where \( \delta \) is the given error threshold); (3) \( P_i(X_i,t_i) \) is the last one satisfying the above two conditions.

In the online segmentation and pruning algorithm, a smaller threshold \( \delta \) is applied by the segmentation to quickly identify a new end point. Introduced noise line segments will be filtered out by the pruning process. The tricky part is we must keep the zigzag shape of the end points, so we must remove two adjacent end points at the same time. We have developed a pruning algorithm to keep the optimized PLR segments by checking the previous line segment for pruning at each new end point, which is illustrated in Figure ??.

For online similarity matching, we must define the function \( f(S_1,S_2) \), introduced in Definition 2, to determine if \( S_1 \) and \( S_2 \) have the same internal structure. First, due to the zigzag shape of the PLR segments, similar subsequences must have the same order of UP, DOWN or NOTREND trends. Second, the relative position of the upper and lower end points must be the same. We use the permutation of a stock PLR stream to satisfy these requirements. The permutation of a subsequence \( S \) with \( n \) elements is a permutation of \( 1, 2, ..., n \). It is calculated through the following steps. Consider a stream of end points: \( S = \{(X_1,t_1), (X_2,t_2), ..., (X_n,t_n)\} \). First, we divide the end points into three subsets by putting all the upper points, lower end points, and no-trend end points into separate subsets. In each subset, the end points are still in the order of time. Without loss of generality, suppose that \( (X_1,t_1) \) is an upper point, \( n \) is even,
and no no-trend points, we will get a new sequence of the \( n \) points as

\[
S' = \{ \{ (X_1, t_1), (X_3, t_3), ..., (X_{n-1}, t_{n-1}) \} \mid \{ (X_2, t_2), (X_4, t_4), ..., (X_n, t_n) \} \}
\]

Next we sort the X values of each subset. We will get another subsequence

\[
S'' = \{ [(X_{i_1}, t_{i_1}), (X_{i_3}, t_{i_3}), ..., (X_{i_{n-1}}, t_{i_{n-1}})] \mid [(X_{i_2}, t_{i_2}), (X_{i_4}, t_{i_4}), ..., (X_{i_n}, t_{i_n})] \}
\]

where \( X_{i_1} \leq X_{i_3} \leq ... \leq X_{i_{n-1}}, X_{i_2} \leq X_{i_4} \leq ... \leq X_{i_n} \) and \( i_1, i_3, ..., i_{n-1} \) is a permutation of 1, 3, ..., \( n-1 \) and \( i_2, i_4, ..., i_n \) is a permutation of 2, 4, ..., \( n \). \( \{ i_1, i_3, ..., i_{n-1}, i_2, i_4, ..., i_n \} \) is called the permutation of \( S \). It represents the relative positions of the upper end points and the lower end points. If two subsequences have the same permutation, they have the same internal structure, \( f(S, S') \) in Definition 2 will be true.

In financial data analysis, we have used simple statistical information to predict the trend at the query event. Our statistical approach is simply to count how many UP, DOWN, NOTREND end points. If there are a large number of similar subsequences at an event, its trend can be predicted based on the frequency of each trend.

### 3.3 Tumor respiratory motion characterization

Another application we have considered concerns the motion of tumor breathing. Two big challenges for tumor motion [29] are (i) the resulting PLR segments should fit the natural understanding of patient breathing stages and (ii) the model must be able to handle irregular breathing which can occur at any time. We propose a PLM using three regular breathing states: exhale (EX), end-of-exhale (EOE) and inhale (IN), and one
irregular breathing state (IRR), as shown in Figure 7b. Motion proceeds from state to state in a fixed order: \(\ldots \Rightarrow EX \Rightarrow EOE \Rightarrow IN \Rightarrow EX \Rightarrow \ldots\). Each of these states corresponds to a natural action: EX is the motion due to lung deflation, EOE is the motion for rest after lung deflation, and IN is the motion due to lung expansion. So in the PLM for tumor motion, \(Q = \{EX, EOE, IN, IRR\}\); \(\Sigma\) is a finite set of real numbers, and \(D\) is illustrated in Figure 7b.

During online segmentation, there are two line segments in the segmentation window, which are important in the state transitions. One is the most recent line segment \(L_r\), as shown in Figure 9a. \(L_r\) is a least square fit of all the data points after last state transition point, and represents the tumor’s current moving state. The other is the decision line segment \(L_d\), which is generated when a new data point arrives. It is a short directed line segment fitted to the \(K\) most recent points. These two line segments and their velocities are used to determine whether or not a new state has occurred.

Based on the piecewise linear model of tumor motion, two sequences have the same internal structure if and only if they have the same order of breathing states. So the function \(f(S_1, S_2)\) in Definition 2 will be true if \(S_1\) and \(S_2\) have the same state order.

Patient similarity provides a convenient way to clustering patients based on their similarity distances and to identify patient features which are correlated with tumor movement. Patient distance can also be used for prediction with clustering. After clustering, if a patient has a new treatment session, then the subsequence similarity matching will only retrieve subsequences from the same cluster, not from patients in other clusters.

4 Future Work

We have proposed a general framework for structured time series stream data analysis, including modeling, piecewise linear segmentation, online similarity matching, and statistical result analysis. We have applied this framework for several applications, including stock market analysis and tumor respiratory motion analysis. We have used the framework for stock data analysis and trend prediction, which is summarized in my SIGMOD 2004 paper [28]. In tumor respiratory motion analysis, the framework is used for quantified motion characterization and prediction in real-time radiation treatment, the results are in the PMB paper [29] and SIGMOD 2005 paper [27].

There are several ongoing research projects based on my previous work, which is briefly summarized in the following:

- One project is to continue working on the stock market data analysis, including discovery of new patterns and clustering data streams based on prediction results. As the same time, we want to incorporate our prediction methods into real systems, which need special concerns about decision making in real time, money management, and special handling in unexpected conditions.

- The other project is based on the previous respiratory motion analysis. We are working on a comprehensive motion prediction approaches based on the finite state model we proposed. The prediction uses dynamic probability distribution, which are optimized by subsequence similarity matching.

- Another project is dynamic data stream clustering for real-time systems. So we can using the clustered information for subsequence similarity matching and trend prediction.

- There are a lot of parameters in our methods. For better results, there should be auto-tuning facilities available that learn the proper settings from training data, and the constants could even adapt themselves during online operation.

- Subsequence similarity matching and building of dynamic probability distribution are very time consuming over very large stream database systems. We are working on access methods to indexing
the historical information of streaming data. The new access methods should be able to well handle frequent mass insertion and deletion and can support efficient range queries as well as point queries.

Table 1 contains a proposed timeline for my dissertation. The schedule is given under the assumption that I will graduate by August 2005.

<table>
<thead>
<tr>
<th>Areas</th>
<th>Subjects</th>
<th>From</th>
<th>To</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tumor motion prediction</td>
<td>Prediction with dynamic probability distribution and optimized by</td>
<td>now</td>
<td>Feb 28, 2005</td>
<td>A manuscript in February 2005</td>
</tr>
<tr>
<td></td>
<td>subsequence similarity matching based on a finite state model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stock data analysis</td>
<td>stock market management and new pattern discovery</td>
<td>March 2005</td>
<td>April 2005</td>
<td>A manuscript in May 2005</td>
</tr>
<tr>
<td></td>
<td>Data stream clustering based on prediction results</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic clustering of data streams</td>
<td>Event-driven dynamic clustering of data streams for real-time applications</td>
<td>April 2005</td>
<td>May 2005</td>
<td>A manuscript in July 2005</td>
</tr>
<tr>
<td></td>
<td>Subsequence similarity matching bases on dynamic clustering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indexing</td>
<td>Streaming data access methods</td>
<td>May 2005</td>
<td>July 2005</td>
<td></td>
</tr>
<tr>
<td>Ph.D. Thesis</td>
<td>Summary and Conclusion</td>
<td></td>
<td></td>
<td>Thesis defense</td>
</tr>
</tbody>
</table>

References


