Sādhanā Vol. 31, Part 2, April 2006, pp. 173–198. © Printed in India

A survey of temporal data mining

SRIVATSAN LAXMAN and P S SASTRY

Department of Electrical Engineering, Indian Institute of Science,
Bangalore 560 012, India
e-mail: {srivats, sastry}@ee.iisc.ernet.in

Abstract. Data mining is concerned with analysing large volumes of (often unstructured) data to automatically discover interesting regularities or relationships which in turn lead to better understanding of the underlying processes. The field of temporal data mining is concerned with such analysis in the case of ordered data streams with temporal interdependencies. Over the last decade many interesting techniques of temporal data mining were proposed and shown to be useful in many applications. Since temporal data mining brings together techniques from different fields such as statistics, machine learning and databases, the literature is scattered among many different sources. In this article, we present an overview of techniques of temporal data mining. We mainly concentrate on algorithms for pattern discovery in sequential data streams. We also describe some recent results regarding statistical analysis of pattern discovery methods.

Keywords. Temporal data mining; ordered data streams; temporal interdependency; pattern discovery.

1. Introduction

Data mining can be defined as an activity that extracts some new nontrivial information contained in large databases. The goal is to discover hidden patterns, unexpected trends or other subtle relationships in the data using a combination of techniques from machine learning, statistics and database technologies. This new discipline today finds application in a wide and diverse range of business, scientific and engineering scenarios. For example, large databases of loan applications are available which record different kinds of personal and financial information about the applicants (along with their repayment histories). These databases can be mined for typical patterns leading to defaults which can help determine whether a future loan application must be accepted or rejected. Several terabytes of remote-sensing image data are gathered from satellites around the globe. Data mining can help reveal potential locations of some (as yet undetected) natural resources or assist in building early warning systems for ecological disasters like oil slicks etc. Other situations where data mining can be of use include analysis of medical records of hospitals in a town to predict, for example, potential outbreaks of infectious diseases, analysis of customer transactions for market research applications etc. The list of application areas for data mining is large and is bound to grow rapidly in the years to come. There are many recent books that detail generic techniques for data mining and discuss various applications (Witten & Frank 2000; Han & Kamber 2001; Hand *et al* 2001).

Temporal data mining is concerned with data mining of large sequential data sets. By sequential data, we mean data that is ordered with respect to some index. For example, time series constitute a popular class of sequential data, where records are indexed by time. Other examples of sequential data could be text, gene sequences, protein sequences, lists of moves in a chess game etc. Here, although there is no notion of time as such, the ordering among the records is very important and is central to the data description/modelling.

Time series analysis has quite a long history. Techniques for statistical modelling and spectral analysis of real or complex-valued time series have been in use for more than fifty years (Box *et al* 1994; Chatfield 1996). Weather forecasting, financial or stock market prediction and automatic process control have been some of the oldest and most studied applications of such time series analysis (Box *et al* 1994). Time series matching and classification have received much attention since the days speech recognition research saw heightened activity (Juang & Rabiner 1993; O'Shaughnessy 2000). These applications saw the advent of an increased role for machine learning techniques like Hidden Markov Models and time-delay neural networks in time series analysis.

Temporal data mining, however, is of a more recent origin with somewhat different constraints and objectives. One main difference lies in the size and nature of data sets and the manner in which the data is collected. Often temporal data mining methods must be capable of analysing data sets that are prohibitively large for conventional time series modelling techniques to handle efficiently. Moreover, the sequences may be nominal-valued or symbolic (rather than being real or complex-valued), rendering techniques such as autoregressive moving average (ARMA) or autoregressive integrated moving average (ARIMA) modelling inapplicable. Also, unlike in most applications of statistical methods, in data mining we have little or no control over the data gathering process, with data often being collected for some entirely different purpose. For example, customer transaction logs may be maintained from an auditing perspective and data mining would then be called upon to analyse the logs for estimating customer buying patterns.

The second major difference (between temporal data mining and classical time series analysis) lies in the kind of information that we want to estimate or unearth from the data. The scope of temporal data mining extends beyond the standard forecast or control applications of time series analysis. Very often, in data mining applications, one does not even know which variables in the data are expected to exhibit any correlations or causal relationships. Furthermore, the exact model parameters (e.g. coefficients of an ARMA model or the weights of a neural network) may be of little interest in the data mining context. Of greater relevance may be the unearthing of useful (and often unexpected) trends or patterns in the data which are much more readily interpretable by and useful to the data owner. For example, a time-stamped list of items bought by customers lends itself to data mining analysis that could reveal which combinations of items tend to be frequently consumed together, or whether there has been some particularly skewed or abnormal consumption pattern this year (as compared to previous years), etc.

In this paper, we provide a survey of temporal data mining techniques. We begin by clarifying the terms models and patterns as used in the data mining context, in the next section. As stated earlier, the field of data mining brings together techniques from machine learning, pattern recognition, statistics etc., to analyse large data sets. Thus many problems and techniques of temporal data mining are also well studied in these areas. Section 3. provides a rough categorization of temporal data mining tasks and presents a brief overview of some of the temporal data mining methods which are also relevant in these other areas. Since these are well-known techniques, they are not discussed in detail. Then, § 4. considers in some detail, the problem of pattern discovery from sequential data. This can be called the quintessential temporal data mining problem. We explain two broad classes of algorithms and also point to many recent developments in this area and to some applications. Section 5. provides a survey of some recent results concerning statistical analysis of pattern discovery methods. Finally, in § 6. we conclude.

2. Models and patterns

The types of structures data mining algorithms look for can be classified in many ways (Han & Kamber 2001; Witten & Frank 2000; Hand *et al* 2001). For example, it is often useful to categorize outputs of data mining algorithms into models and patterns (Hand *et al* 2001, chapter 6). Models and patterns are structures that can be estimated from or matched for in the data. These structures may be utilized to achieve various data mining objectives.

A *model* is a global, high-level and often abstract representation for the data. Typically, models are specified by a collection of model parameters which can be estimated from the given data. Often, it is possible to further classify models based on whether they are predictive or descriptive. Predictive models are used in forecast and classification applications while descriptive models are useful for data summarization. For example, autoregression analysis can be used to guess future values of a time series based on its past. Markov models constitute another popular class of predictive models that has been extensively used in sequence classification applications. On the other hand, spectrograms (obtained through time-frequency analysis of time series) and clustering are good examples of descriptive modelling techniques. These are useful for data visualization and help summarize data in a convenient manner.

In contrast to the (global) model structure, a *pattern* is a local structure that makes a specific statement about a few variables or data points. Spikes, for example, are patterns in a real-valued time series that may be of interest. Similarly, in symbolic sequences, regular expressions constitute a useful class of well-defined patterns. In biology, genes, regarded as the classical units of genetic information, are known to appear as local patterns interspersed between chunks of non-coding DNA. Matching and discovery of such patterns are very useful in many applications. Due to their readily interpretable structure, patterns play a particularly dominant role in data mining.

Finally, we note that, while this distinction between models and patterns is useful from the point of view comparing and categorizing data mining algorithms, there are cases when such a distinction becomes blurred. This is bound to happen given the inherent interdisciplinary nature of the data mining field (Smyth 2001). In fact, later in § 5., we discuss examples of how model-based methods can be used to better interpret patterns discovered in data, thereby enhancing the utility of both structures in temporal data mining.

3. Temporal data mining tasks

Data mining has been used in a wide range of applications. However, the possible objectives of data mining, which are often called *tasks* of data mining (Han & Kamber 2001, chapter 4; Hand *et al* 2001, chapter 1) can be classified into some broad groups. For the case of temporal data mining, these tasks may be grouped as follows: (i) prediction, (ii) classification, (iii) clustering,

(iv) search & retrieval and (v) pattern discovery. Once again, as was the case with models and patterns, this categorization is neither unique nor exhaustive, the only objective being to facilitate an easy discussion of the numerous techniques in the field.

Of the five categories listed above, the first four have been investigated extensively in traditional time series analysis and pattern recognition. Algorithms for pattern discovery in large databases, however, are of more recent origin and are mostly discussed only in data mining literature. In this section, we provide a brief overview of temporal data mining techniques as relevant to prediction, classification, clustering and search & retrieval. In the next section, we provide a more detailed account of pattern discovery techniques for sequential data.

3.1 *Prediction*

The task of time-series prediction has to do with forecasting (typically) future values of the time series based on its past samples. In order to do this, one needs to build a predictive model for the data. Probably the earliest example of such a model is due to Yule way back in 1927 (Yule 1927). The autoregressive family of models, for example, can be used to predict a future value as a linear combination of earlier sample values, provided the time series is assumed to be stationary (Box *et al* 1994; Chatfield 1996; Hastie *et al* 2001). Linear nonstationary models like ARIMA models have also been found useful in many economic and industrial applications where some suitable variant of the process (e. g. differences between successive terms) can be assumed to be stationary. Another popular work-around for nonstationarity is to assume that the time series is piece-wise (or locally) stationary. The series is then broken down into smaller "frames" within each of which, the stationarity condition can be assumed to hold and then separate models are learnt for each frame. In addition to these standard ARMA family of models, there are many nonlinear models for time series prediction. For example, neural networks have been put to good use for nonlinear modelling of time series data (Sutton 1988; Wan 1990; Haykin 1992, chapter 13; Koskela *et al* 1996). The prediction problem for symbolic sequences has been addressed in AI research. For example, Dietterich & Michalski (1985) consider various rule models (like disjunctive normal form model, periodic rule model etc.). Based on these models sequence-generating rules are obtained that (although may not completely determine the next symbol) state some properties that constrain which symbol can appear next in the sequence.

3.2 *Classification*

In sequence classification, each sequence presented to the system is assumed to belong to one of finitely many (predefined) classes or categories and the goal is to automatically determine the corresponding category for the given input sequence. There are many examples of sequence classification applications, like speech recognition, gesture recognition, handwritten word recognition, demarcating gene and non-gene regions in a genome sequence, on-line signature verification, etc. The task of a speech recognition system is to transcribe speech signals into their corresponding textual representations (Juang & Rabiner 1993; O'Shaughnessy 2000; Gold & Morgan 2000). In gesture (or human body motion) recognition, video sequences containing hand or head gestures are classified according to the actions they represent or the messages they seek to convey. The gestures or body motions may represent, e. g., one of a fixed set of messages like waving hello, goodbye, and so on (Darrell & Pentland 1993), or they could be the different strokes in a tennis video (Yamato *et al* 1992), or in other cases, they could belong to the dictionary of some sign language (Starner & Pentland 1995) etc. There are some pattern recognition applications in which even images are viewed as sequences. For example, images of handwritten words are sometimes regarded as a sequence of pixel columns or segments proceeding from left to right in the image. Recognizing the words in such sequences is another interesting sequence classification application (Kundu *et al* 1988; Tappert *et al* 1990). In on-line handwritten word recognition (Nag *et al* 1986) and signature verification applications (Nalwa 1997), the input is a sequence of pixel coordinates drawn by the user on a digitized tablet and the task is to assign a pattern label to each sequence.

As is the case with any standard pattern recognition framework (Duda *et al* 1997), in these applications also, there is a feature extraction step that precedes the classification step. For example, in speech recognition, the standard analysis method is to divide the speech pattern into frames and apply a feature extraction method (like linear prediction or mel-cepstral analysis) on each frame. In gesture recognition, motion trajectories and other object-related image features are obtained from the video sequence. The feature extraction step in sequence recognition applications typically generates, for each pattern (such as a video sequence or speech utterance), a sequence of feature vectors that must then be subjected to a classification step.

Over the years, sequence classification applications have seen the use of both pattern-based as well as model-based methods. In a typical pattern-based method, prototype feature sequences are available for each class (i. e. for each word, gesture etc.). The classifier then searches over the space of all prototypes, for the one that is closest (or most similar) to the feature sequence of the new pattern. Typically, the prototypes and the given features vector sequences are of different lengths. Thus, in order to score each prototype sequence against the given pattern, sequence aligning methods like Dynamic Time Warping are needed. We provide a more detailed review of sequence alignment methods and similarity measures later in § 3.4. Another popular class of sequence recognition techniques is a model-based method that use Hidden Markov Models (HMMs). Here, one HMM is learnt from training examples for each pattern class and a new pattern is classified by asking which of these HMMs is most likely to generate it. In recent times, many other model-based methods have been explored for sequence classification. For example, Markov models are now frequently used in biological sequence classification (Baldi *et al* 1994; Ewens & Grant 2001) and financial time-series prediction (Tino *et al* 2000). Machine learning techniques like neural networks have also been used for protein sequence classification (e. g. see Wu *et al* 1995). Haselsteiner & Pfurtscheller (2000) use time-dependent neural network paradigms for EEG signal classification.

3.3 *Clustering*

Clustering of sequences or time series is concerned with grouping a collection of time series (or sequences) based on their similarity. Clustering is of particular interest in temporal data mining since it provides an attractive mechanism to automatically find some structure in large data sets that would be otherwise difficult to summarize (or visualize). There are many applications where a time series clustering activity is relevant. For example in web activity logs, clusters can indicate navigation patterns of different user groups. In financial data, it would be of interest to group stocks that exhibit similar trends in price movements. Another example could be clustering of biological sequences like proteins or nucleic acids so that sequences within a group have similar functional properties (Corpet 1988; Miller *et al* 1999; Osata *et al* 2002). There are a variety of methods for clustering sequences. At one end of the spectrum, we have model-based sequence clustering methods (Smyth 1997; Sebastiani *et al* 1999; Law & Kwok 2000). Learning mixture models, for example, constitute a big class of model-based clustering methods. In case of time series clustering, mixtures of, e. g., ARMA models (Xiong & Yeung 2002) or Hidden Markov Models (Cadez *et al* 2000; Alon *et al* 2003) are in popular use. The other broad class in sequence clustering uses pattern alignment-based scoring (Corpet 1988; Fadili *et al* 2000) or similarity measures (Schreiber & Schmitz 1997; Kalpakis & Puttagunta 2001) to compare sequences. The next section discusses similarity measures in some more detail. Some techniques use both model-based as well as alignment-based methods (Oates *et al* 2001).

3.4 *Search and retrieval*

Searching for sequences in large databases is another important task in temporal data mining. Sequence search and retrieval techniques play an important role in interactive explorations of large sequential databases. The problem is concerned with efficiently locating subsequences (often referred to as queries) in large archives of sequences (or sometimes in a single long sequence). Query-based searches have been extensively studied in language and automata theory . While the problem of efficiently locating exact matches of (some well-defined classes of) substrings is well solved, the situation is quite different when looking for *approximate* matches (Wu & Manber 1992). In typical data mining applications like content-based retrieval, it is approximate matching that we are more interested in.

In content-based retrieval, a query is presented to the system in the form of a sequence. The task is to search a (typically) large data base of sequential data and retrieve from it sequences or subsequences *similar* to the given query sequence. For example, given a large music database the user could "hum" a query and the system should retrieve tracks that resemble it (Ghias *et al* 1995). In all such problems there is a need to quantify the extent of similarity between any two (sub)sequences. Given two sequences of equal length we can define a measure of similarity by considering distances between corresponding elements of the two sequences. The individual elements of the sequences may be vectors of real numbers (e. g. in applications involving speech or audio signals) or they may symbolic data (e. g. in applications involving gene sequences). When the sequence elements are feature vectors (with real components) standard metrics such as Euclidean distance may be used for measuring similarity between two elements. However, sometimes the Euclidean norm is unable to capture subjective similarities effectively. For example, in speech or audio signals, similar sounding patterns may give feature vectors that have large Euclidean distances and vice versa. An elaborate treatment of distortion measures for speech and audio signals (e. g. log spectral distances, weighted cepstral distances, etc.) can be found in (Gray *et al* 1980; Juang & Rabiner 1993, chapter 4). The basic idea in these measures is to perform the comparison in spectral domain by emphasizing differences in those spectral components that are perceptually more relevant. Similarity measures based on other transforms have been explored as well. For example, Wu *et al* (2000) present a comparison of DFT and DWT-based similarity searches. Perng *et al* (2000) propose similarity measures which are invariant under various transformations (like shifting, amplitude scaling etc.). When the sequences consist of symbolic data we have to define dissimilarity between every pair of symbols which in general is determined by the application (e. g. PAM and BLOSUM have been designed by biologists for aligning amino acid sequences (Gusfield 1997; Ewens & Grant 2001)).

Choice of similarity or distortion measure is only one aspect of the sequence matching problem. In most applications involving determination of similarity between pairs of sequences, the sequences would be of different lengths. In such cases, it is not possible to blindly accumulate distances between corresponding elements of the sequences. This brings us to the second aspect of sequence matching, namely, sequence alignment. Essentially we need to properly insert 'gaps' in the two sequences or decide which should be corresponding elements in the two sequences. Time warping methods have been used for sequence classification and matching for many years (Kruskal 1983; Juang & Rabiner 1993, chapter 4; Gold & Morgan 2000). In speech applications, Dynamic Time Warping (DTW) is a systematic and efficient method (based on dynamic programming) that identifies which correspondence among feature vectors of two sequences is best when scoring the similarity between them. In recent times, DTW and its variants are being used for motion time series matching (Chang *et al* 1998; Sclaroff *et al* 2001) in video sequence mining applications as well. DTW can, in general, be used for sequence alignment even when the sequences consist of symbolic data. There are many situations in which such symbolic sequence matching problems find applications. For example, many biological sequences such as genes, proteins, etc., can be regarded as sequences over a finite alphabet. When two such sequences are similar, it is expected that the corresponding biological entities have similar functions because of related biochemical mechanisms (Frenkel 1991; Miller *et al* 1994). Many problems in bioinformatics relate to the comparison of DNA or protein sequences, and time-warping-based alignment methods are well suited for such problems (Ewens & Grant 2001; Cohen 2004). Two symbolic sequences can be compared by defining a set of "edit" operations (Durbin *et al* 1998; Levenshtein 1966), namely symbol insertion, deletion and substitution, together with a cost for each such operation. Each "warp" in the DTW sense, corresponds to a sequence of edit operations. The distance between two strings is defined as the least sum of edit operation costs that needs to be incurred when comparing them.

Another approach that has been used in time series matching is to regard two sequences as similar if they have enough non-overlapping time-ordered pairs of subsequences that are similar. This idea was applied to find matches in a US mutual fund database (Agrawal *et al* 1995a). In some applications it is possible to locally estimate some symbolic features (e. g. local shapes in signal waveforms) in real-valued time series and match the corresponding symbolic sequences (Agrawal *et al* 1995b). Approaches like this are particularly relevant for data mining applications since there is considerable efficiency to be gained by reducing the data from real-valued time series to symbolic sequences, and by performing the sequence matching in this new higher level of abstraction. Recently, Keogh & Pazzani (2000) used a piece-wise aggregate model for time-series to allow faster matching using dynamic time warping. There is a similar requirement for sequence alignment when comparing symbolic sequences too (Gusfield 1997).

4. Pattern discovery

Previous sections introduced the idea of patterns in sequential data and in particular § 3.4 described how patterns are typically matched and retrieved from large sequential data archives. In this section we consider the temporal data mining task of pattern discovery. Unlike in search and retrieval applications, in pattern discovery there is no specific query in hand with which to search the database. The objective is simply to unearth all *patterns of interest*. It is worthwhile to note at this point that whereas the other temporal data mining tasks discussed earlier in § 3. (i. e. sequence prediction, classification, clustering and matching) had their origins in other disciplines like estimation theory, machine learning or pattern recognition, the pattern discovery task has its origins in data mining itself. In that sense, pattern discovery, with its exploratory and unsupervised nature of operation, is something of a sole preserve of data mining. For this reason, this review lays particular emphasis on the temporal data mining task of pattern discovery.

In this section, we first introduce the notion of frequent patterns and point out its relevance to rule discovery. Then we discuss, at some length, two popular frameworks for frequent pattern discovery, namely sequential patterns and episodes. In each case we explain the basic algorithm and then state some recent improvements. We end the section by discussing another important pattern class, namely, partially periodic patterns.

As mentioned earlier, a pattern is a local structure in the data. It would typically be like a substring or a substring with some 'don't care' characters in it etc. The problem of pattern discovery is to unearth all 'interesting' patterns in the data. There are many ways of defining what constitutes a pattern and we shall discuss some generic methods of defining patterns which one can look for in the data. There is no universal notion for interestingness of a pattern either. However, one concept that is found very useful in data mining is that of frequent patterns. A frequent pattern is one that occurs many times in the data. Much of data mining literature is concerned with formulating useful pattern structures and developing efficient algorithms for discovering all patterns which occur frequently in the data.

Methods for finding frequent patterns are considered important because they can be used for discovering useful rules. These rules can in turn be used to infer some interesting regularities in the data. A *rule* consists of a pair of Boolean-valued propositions, namely, a left-hand side proposition (the antecedent) and a right-hand side proposition (the consequent). The rule states that when the antecedent is true, then the consequent will be true as well. Rules have been popular representations of knowledge in machine learning and AI for many years. Decision tree classifiers, for example, yield a set of classification rules to categorize data. In data mining, *association* rules are used to capture correlations between different attributes in the data (Agrawal & Srikant 1994). In such cases, the (estimate of) conditional probability of the consequent occurring given the antecedent, is referred to as *confidence* of the rule. For example, in a sequential data stream, if the pattern "B follows A" appears f_1 times and the pattern "C follows B follows A" appears f_2 times, it is possible to infer a temporal association rule "whenever B follows A, C will follow too" with a confidence (f_2/f_1). A rule is usually of interest, only if it has high confidence and it is applicable sufficiently often in the data, i. e., in addition to the confidence (f_2/f_1) being high, frequency of the consequent (f_2) should also be high.

One of the earliest attempts at discovering patterns (of sufficiently general interest) in sequential databases is a pattern discovery method for a large collection of protein sequences (Wang *et al* 1994). A protein is essentially a sequence of amino acids. There are 20 amino acids that commonly appear in proteins, so that, by denoting each amino acid by a distinct letter, it is possibly to describe proteins (for computational purposes) as symbolic sequences over an alphabet of size twenty. As was mentioned earlier, protein sequences that are similar or those that share similar subsequences are likely to perform similar biological functions.

Wang *et al* (1994) consider a large database of more than 15000 protein sequences. Biologically related (and functionally similar) proteins are grouped together into around 700 groups. The problem now is to search for representative (temporal) patterns within each group. Each temporal pattern is of the form $*X_1 * X_2 \cdots * X_N$ where the X_i's are the symbols defining the pattern and $*$ denotes a variable length "don't care" sequence. A pattern is considered to be of interest if it is sufficiently long and approximately matches sufficiently many protein sequences in the database. The minimum length and minimum number of matches are user-defined parameters. The method by Wang *et al* (1994) first finds some candidate segments by constructing a generalized suffix tree for a small sample of the sequences from the full database. These are then combined to construct candidate patterns and the full database is then searched for each of these candidate patterns using an edit distance based scoring scheme.

The number of sequences (in the database) which are within some user-defined distance of a given candidate pattern is its final occurrence score and those patterns whose score exceeds a user-defined threshold are the output temporal patterns. These constitute the representative patterns (referred to here as *motifs*) for the proteins within a group. The motifs so discovered in each protein group are used as templates for the group in a sequence classifier application. The underlying pattern discovery method described by Wang *et al* (1994) however, is not guaranteed to be complete (in the sense that, given a set of sequences, it may not discover *all* the temporal patterns in the set that meet the user-defined threshold constraints). A complete solution to a similar, and in fact, a more general formulation of this problem is presented by Agrawal & Srikant (1995) in the context of data mining of a large collection of customer transaction sequences. This can, arguably, be regarded as the birth of the field of temporal data mining. We discuss this approach to sequential pattern mining in the subsection below.

4.1 *Sequential patterns*

The framework of sequential pattern discovery is explained here using the example of a customer transaction database as by Agrawal & Srikant (1995). The database is a list of time-stamped transactions for each customer that visits a supermarket and the objective is to discover (temporal) buying patterns that sufficiently many customers exhibit. This is essentially an extension (by incorporation of temporal ordering information into the patterns being discovered) of the original association rule mining framework proposed for a database of unordered transaction records (Agrawal *et al* 1993) which is known as the *Apriori* algorithm. Since there are many temporal pattern discovery algorithms that are modelled along the same lines as the Apriori algorithm, it is useful to first understand how Apriori works before discussing extensions to the case of temporal patterns.

Let $\mathcal{D}$ be a database of customer transactions at a supermarket. A transaction is simply an unordered collection of items purchased by a customer in one visit to the supermarket. The Apriori algorithm systematically unearths all patterns in the form of (unordered) sets of items that appear in a sizable number of transactions. We introduce some notation to precisely define this framework. A non-empty set of items is called an *itemset*. An itemset i is denoted by $(i_1 i_2 \cdots i_m)$, where i_j is an item. Since i has m items, it is sometimes called an m-itemset. Trivially, each transaction in the database is an itemset. However, given an arbitrary itemset i, it may or may not be contained in a given transaction T. The fraction of all transactions in the database in which an itemset is contained in is called the *support* of that itemset. An itemset whose support exceeds a user-defined threshold is referred to as a *frequent* itemset. These itemsets are the patterns of interest in this problem. The brute force method of determining supports for all possible itemsets (of size m for various m) is a combinatorially explosive exercise and is not feasible in large databases (which is typically the case in data mining). The problem therefore is to find an efficient algorithm to discover all frequent itemsets in the database $\mathcal{D}$ given a user-defined minimum support threshold.

The Apriori algorithm exploits the following very simple (but amazingly useful) principle: if i and j are itemsets such that j is a subset of i, then the support of j is greater than or equal to the support of i. Thus, for an itemset to be frequent all its subsets must in turn be frequent as well. This gives rise to an efficient level-wise construction of frequent itemsets in $\mathcal{D}$. The algorithm makes multiple passes over the data. Starting with itemsets of size 1 (i. e. 1-itemsets), every pass discovers frequent itemsets of the next bigger size. The first pass over the data discovers all the frequent 1-itemsets. These are then combined to generate candidate 2-itemsets and by determining their supports (using a second pass over the data) the frequent 2-itemsets are found. Similarly, these frequent 2-itemsets are used to first obtain candidate

3-itemsets and then (using a third database pass) the frequent 3-itemsets are found, and so on. The candidate generation before the m^{th} pass uses the Apriori principle described above as follows: an m-itemset is considered a candidate only if all $(m-1)$-itemsets contained in it have already been declared frequent in the previous step. As m increases, while the number of all possible m-itemsets grows exponentially, the number of frequent m-itemsets grows much slower, and as a matter of fact, starts decreasing after some m. Thus the candidate generation method in Apriori makes the algorithm efficient. This process of progressively building itemsets of the next bigger size is continued till a stage is reached when (for some size of itemsets) there are no frequent itemsets left to continue. This marks the end of the frequent itemset discovery process.

We now return to the sequential pattern mining framework of Agrawal & Srikant (1995) which basically extends the frequent itemsets idea described above to the case of patterns with temporal order in them. The database $\mathcal{D}$ that we now consider is no longer just some unordered collection of transactions. Now, each transaction in $\mathcal{D}$ carries a time-stamp as well as a customer ID. Each transaction (as earlier) is simply a collection of items. The transactions associated with a single customer can be regarded as a sequence of itemsets (ordered by time), and $\mathcal{D}$ would have one such transaction sequence corresponding to each customer. In effect, we have a database of transaction sequences, where each sequence is a list of transactions ordered by transaction-time.

Consider an example database with 5 customers whose corresponding transaction sequences are as follows: (1) $\langle (AB)\ (ACD)\ (BE)\rangle$, (2) $\langle (D)\ (ABE)\rangle$, (3) $\langle (A)\ (BD)\ (ABEF)\ (GH)\rangle$, (4) $\langle (A)\ (F)\rangle$, and (5) $\langle (AD)\ (BEGH)\ (F)\rangle$. Here, each customer's transaction sequence is enclosed in angular braces, while the items bought in a single transaction are enclosed in round braces. For example, customer 3 made 4 visits to the supermarket. In his first visit he bought only item A, in the second he bought items B and D, and so on.

The temporal patterns of interest are also essentially some (time ordered) sequences of itemsets. A sequence s of itemsets is denoted by $\langle s_1\ s_2 \cdots s_n\rangle$, where s_j is an itemset. Since s has n itemsets, it is called an n-sequence. A sequence $a = \langle a_1\ a_2 \cdots a_n\rangle$ is said to be *contained in* another sequence $b = \langle b_1\ b_2 \cdots b_m\rangle$ (or alternately, b is said to contain a) if there exist integers $i_1 < i_2 < \cdots < i_n$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots , a_n \subseteq b_{i_n}$. That is, an n-sequence a is contained in a sequence b if there exists an n-length subsequence in b, in which each itemset contains the corresponding itemsets of a. For example, the sequence $\langle (A)(BC)\rangle$ is contained in $\langle (AB)\ (F)\ (BC)\ (DE)\rangle$ but not in $\langle (BC)\ (AB)\ (C)\ (DE)\rangle$. Further, a sequence is said to be *maximal* in a set of sequences, if it is not contained in any other sequence. In the set of example customer transaction sequences listed above, all are maximal (with respect to this set of sequences) except the sequence of customer 4, which is contained in, e. g., transaction sequence of customer 3.

The *support* for any arbitrary sequence, a, of itemsets, is the fraction of customer transaction sequences in the database $\mathcal{D}$ which contain a. For our example database, the sequence $\langle (D)(GH)\rangle$ has a support of 0.4, since it is contained in 2 of the 5 transaction sequences (namely that of customer 3 and customer 5). The user specifies a minimum support threshold. Any sequence of itemsets with support greater than or equal to this threshold is called a *large* sequence. If a sequence a is large and maximal (among the set of all large sequences), then it is regarded as a *sequential pattern*. The task is to systematically discover all sequential patterns in $\mathcal{D}$.

While we described the framework using an example of mining a database of customer transaction sequences for temporal buying patterns, this concept of sequential patterns is quite general and can be used in many other situations as well. Indeed, the problem of motif discovery in a database of protein sequences that was discussed earlier can also be easily addressed in this framework. Another example is web navigation mining. Here the database contains a sequence of websites that a user navigates through in each browsing session. Sequential pattern mining can be used to discover those sequences of websites that are frequently visited one after another.

We next discuss the mechanism of sequential pattern discovery. The search for sequential patterns begins with the discovery of all possible itemsets with sufficient support. The Apriori algorithm described earlier can be used here, except that there is a small difference in the definition of support. Earlier, the support of an itemset was defined as the fraction of *all* transactions that contained the itemset. But here, the support of an itemset is the fraction of *customer transaction sequences* in which at least one transaction contains the itemset. Thus, a frequent itemset is essentially the same as a large 1-sequence (and so is referred to as a *large* itemset or *litemset*) . Once all litemsets in the data are found, a transformed database is obtained where, within each customer transaction sequence, each transaction is replaced by the litemsets contained in that transaction.

The next step is called the sequence phase, where again, multiple passes are made over the data. Before each pass, a set of new potentially large sequences called candidate sequences are generated. Two families of algorithms are presented by Agrawal & Srikant (1995) and are referred to as count-all and count-some algorithms. The count-all algorithm first counts all the large sequences and then prunes out the non-maximal sequences in a post-processing step. This algorithm is again based on the general idea of the Apriori algorithm of Agrawal & Srikant (1994) for counting frequent itemsets. In the first pass through the data the large 1-sequences (same as the litemsets) are obtained. Then candidate 2-sequences are constructed by combining large 1-sequences with litemsets in all possible ways. The next pass identifies the large 2-sequences. Then large 3-sequences are obtained from large 2-sequences, and so on.

The count-some algorithms by Agrawal & Srikant (1995) intelligently exploit the maximality constraint. Since the search is only for maximal sequences, we can avoid counting sequences which would anyways be contained in longer sequences. For this we must count longer sequences first. Thus, the count-some algorithms have a forward phase, in which all frequent sequences of certain lengths are found, and then a backward phase, in which all the remaining frequent sequences are discovered. It must be noted however, that if we count a lot of longer sequences that do not have minimum support, the efficiency gained by exploiting the maximality constraint, may be offset by the time lost in counting sequences without minimum support (which of course, the count-all algorithm would never have counted because their subsequences were not large). These sequential pattern discovery algorithms are quite efficient and are used in many temporal data mining applications and are also extended in many directions.

The last decade has seen many sequential pattern mining methods being proposed from the point of view of improving upon the performance of the algorithm by Agrawal & Srikant (1995). Parallel algorithms for efficient sequential pattern discovery are proposed by Shintani & Kitsuregawa (1998). The algorithms by Agrawal & Srikant (1995) need as many database passes as the length of the longest sequential pattern. Zaki (1998) proposes a lattice-theoretic approach to decompose the original search space into smaller pieces (each of which can be independently processed in main-memory) using which the number of passes needed is reduced considerably. Lin & Lee (2003) propose a system for interactive sequential pattern discovery, where the user queries with several minimum support thresholds iteratively and discovers the desired set of patterns corresponding to the last threshold.

Another class of variants of the sequential pattern mining framework seek to provide extra user-controlled focus to the mining process. For example, Srikanth & Agrawal (1996) generalize the sequential patterns framework to incorporate some user-defined taxonomy of items as well as minimum and maximum time-interval constraints between elements in a sequence. *Constrained association queries* are proposed (Ng *et al* 1998) where the user may specify some domain, class and aggregate constraints on the rule antecedents and consequents. Recently, a family of algorithms called SPIRIT (Sequential Pattern mIning with Regular expressIon consTraints) is proposed in order to mine frequent sequential patterns that also belong to the language specified by the user-defined regular expressions (Garofalakis *et al* 2002).

The performance of most sequential pattern mining algorithms suffers when the data has long sequences with sufficient support, or when using very low support thresholds. One way to address this issue is to search, not just for large sequences (i. e. those with sufficient support), but for sequences that are *closed* as well. A large sequence is said to be *closed* if it is not properly contained in any other sequence which has the same support. The idea of mining data sets for frequent *closed* itemsets was introduced by Pasquier *et al* (1999). Techniques for mining sequential closed patterns are proposed by Yan *et al* (2003); Wang & Han (2004). The algorithm by Wang & Han (2004) is particularly interesting in that it presents an efficient method for mining sequential closed patterns without an explicit iterative candidate generation step.

4.2 *Frequent episodes*

A second class of approaches to discovering temporal patterns in sequential data is the frequent episode discovery framework (Mannila *et al* 1997). In the sequential patterns framework, we are given a collection of sequences and the task is to discover (ordered) sequences of items (i. e. sequential patterns) that occur in sufficiently many of those sequences. In the frequent episodes framework, the data are given in a single long sequence and the task is to unearth temporal patterns (called episodes) that occur sufficiently often along that sequence.

Mannila *et al* (1997) apply frequent episode discovery for analysing alarm streams in a telecommunication network. The status of such a network evolves dynamically with time. There are different kinds of alarms that are triggered by different states of the telecommunication network. Frequent episode mining can be used here as part of an alarm management system. The goal is to improve understanding of the relationships between different kinds of alarms, so that, e. g., it may be possible to foresee an impending network congestion, or it may help improve efficiency of the network management by providing some early warnings about which alarms often go off close to one another. We explain below the framework of frequent episode discovery.

The data, referred to here as an *event sequence*, are denoted by $\langle (E_1, t_1), (E_2, t_2), \ldots \rangle$, where E_i takes values from a finite set of event types $\mathcal{E}$, and t_i is an integer denoting the time stamp of the ith event. The sequence is ordered with respect to the time stamps so that, $t_i \le t_{i+1}$ for all $i = 1, 2, \ldots$. The following is an example event sequence with 10 events in it:

$$\langle (A, 2), (B, 3), (A, 7), (C, 8), (B, 9), (D, 11), (C, 12), (A, 13), (B, 14), (C, 15) \rangle. \qquad (1)$$

An episode α is defined by a triple $(V_\alpha, \le_\alpha, g_\alpha)$, where V_α is a collection of nodes, $\le_\alpha$ is a partial order on V_α and $g_\alpha : V_\alpha \to \mathcal{E}$ is a map that associates each node in the episode with an event type. Put in simpler terms, an episode is just a partially ordered set of event types. When the order among the event types of an episode is total, it is called a *serial* episode and when there is no order at all, the episode is called a *parallel* episode. For example, $(A \rightarrow B \rightarrow C)$ is a 3-node serial episode. The arrows in our notation serve to emphasize the total order. In contrast, parallel episodes are somewhat similar to itemsets, and so, we can denote a 3-node parallel episode with event types A, B and C, as (ABC). Although, one can have episodes that are neither serial nor parallel, the episode discovery framework of Mannila *et al* (1997) is mainly concerned with only these two varieties of episodes.

An episode is said to *occur* in an event sequence if there exist events in the sequence occurring with exactly the same order as that prescribed in the episode. For example, in the example event sequence (1), the events $(A, 2)$, $(B, 3)$ and $(C, 8)$ constitute an occurrence of the serial episode $(A \rightarrow B \rightarrow C)$ while the events $(A, 7)$, $(B, 3)$ and $(C, 8)$ do not, because for this serial episode to occur, A must occur before B and C. Both these sets of events, however, are valid occurrences of the parallel episode (ABC), since there are no restrictions with regard to the order in which the events must occur for parallel episodes.

Recall that in the case of sequential patterns, we defined the notion of when a sequence is contained in another. Similarly here there is the idea of subepisodes. Let α and β be two episodes. β is said to be a *subepisode* of α if all the event types in β appear in α as well, and if the partial order among the event types of β is the same as that for the corresponding event types in α. For example, $(A \rightarrow C)$ is a 2-node subepisode of the serial episode $(A \rightarrow B \rightarrow C)$ while $(B \rightarrow A)$ is not a subepisode. In case of parallel episodes, this order constraint is not there, and so every subset of the event types of an episode correspond to a subepisode.

Finally, in order to formulate a frequent episode discovery framework, we need to fix the notion of episode *frequency*. Once a frequency is defined for episodes (in an event sequence) the task is to efficiently discover all episodes that have frequency above some (user-specified) threshold. For efficiency purposes, one likes to use the basic idea of the Apriori algorithm and hence it is necessary to stipulate that the frequency is defined in such a way that the frequency of an episode is never larger than that of any of its subepisodes. This would ensure that an n-node episode is a candidate frequent episode only if all its $(n-1)$-node subepisodes are frequent. Mannila *et al* (1997) define the frequency of an episode as the fraction of all fixed-width sliding windows over the data in which the episode occurs at least once. Note that if an episode occurs in a window then all its subepisodes occur in it as well. The user specifies the width of the sliding window. Now, given an event sequence, a window-width and a frequency threshold, the task is to discover all frequent episodes in the event sequence. Once the frequent episodes are known, it is possible to generate rules (that describe temporal correlations between events) along the lines described earlier. The rules obtained in this framework would have the "subepisode implies episode" form, and the confidence, as earlier, would be the appropriate ratio of episode frequencies.

This kind of a temporal pattern mining formulation has many interesting and useful application possibilities. As was mentioned earlier, this framework was originally applied to analysing alarm streams in a telecommunication network (Mannila *et al* 1997). Another application is the mining of data from assembly lines in manufacturing plants (Laxman *et al* 2004a). The data are an event sequence that describes the time-evolving status of the assembly line. At any given instant, the line is either running or it is halted due to some reason (like lunch break, electrical problem, hydraulic failure etc.). There are codes assigned for each of these situations and these codes are logged whenever there is a change in the status of the line. This sequence of time-stamped status codes constitutes the data for each line. The frequent episode discovery framework is used to unearth some temporal patterns that could help understanding hidden correlations between different fault conditions and hence improving the performance and throughputs of the assembly line. In manufacturing plants, sometimes it is known that one particular line performs significantly better than another (although no prior reason is attributable to this difference). Here, frequent episode discovery may actually facilitate the devising of some process improvements by studying the frequent episodes in one line and comparing them to those in the other. The frequent episode discovery framework has also been applied on many other kinds of data sets, like web navigation logs (Mannila *et al* 1997; Casas-Garriga 2003), and Wal-Mart sales data (Atallah *et al* 2004) etc.

The process of frequent episode discovery is an Apriori-style level-wise algorithm that starts with discovering frequent 1-node episodes. These are then combined to form candidate 2-node episodes and then by counting their frequencies, 2-node frequent episodes are obtained. This process is continued till frequent episodes of all lengths are found. Like in the Apriori algorithm, the candidate generation step here declares an episode as a candidate only if all its subepisodes have already been found frequent. This kind of a construction of bigger episodes from smaller ones is possible because the definition of episode frequency guarantees that subepisodes are at least as frequent as the episode. Starting with the same set of frequent 1-node episodes, the algorithms for candidate generation differ slightly for the two cases of parallel and serial episodes (due to the extra total order constraint imposed in the latter case). The difference between the two frequency counting algorithms (for parallel and serial episodes) is more pronounced.

Counting frequencies of parallel episodes is comparatively straightforward. As mentioned earlier, parallel episodes are like itemsets and so counting the number of sliding windows in which they occur is much like computing the support of an itemset over a list of customer transactions. An $\mathcal{O}((n + l^2)k)$ algorithm is presented by (Mannila *et al* (1997) for computing the frequencies of a set of k, l-node parallel episodes in an n-length event sequence. Counting serial episodes, on the other hand, is a bit more involved. This is because, unlike for parallel episodes, we need finite state automata to recognize serial episodes. More specifically, an appropriate l-state automaton can be used to recognize occurrences of an l-node serial episode. The automaton corresponding to an episode accepts that episode and rejects all other input. For example, for the episode $(A \rightarrow B \rightarrow C)$, there would be a 3-state automaton that transits to its first state on seeing an event of type A and then waits for an event of type B to transit to its next state and so on. When this automaton transits to its final state, the episode is recognized (to have occurred once) in the event sequence. We need such automata for each episode α whose frequency is being counted. In general, while traversing an event sequence, at any given time, there may be any number of partial occurrences of a given episode and hence we may need any number of different instances of the automata corresponding to this episode to be active if we have to count all occurrences of the episode. Mannila *et al* (1997), present an algorithm which needs only l instances of the automata (for each l-node episode) to be able to obtain the frequency of the episode.

It is noted here that such an automata-based counting scheme is particularly attractive since it facilitates the frequency counting of not one but an entire set of serial episodes in one pass through the data. For a set of k l-node serial episodes the algorithm has $\mathcal{O}(lk)$ space complexity. The corresponding time complexity is given by $\mathcal{O}(nlk)$, where n is, as earlier, the length of the event stream being mined.

The episode discovery framework described so far employs the windows-based frequency measure for episodes (which was proposed by Mannila *et al* 1997). However, there can be other ways to define episode frequency. One such alternative is proposed by Mannila *et al* (1997) itself and is based on counting what are known as *minimal occurrences* of episodes. A minimal occurrence of an episode is defined as a window (or contiguous slice) of the input sequence in which the episode occurs, and further, no proper sub-window of this window contains an occurrence of the episode. The algorithm for counting minimal occurrences trades space efficiency for time efficiency when compared to the windows-based counting algorithm. In addition, since the algorithm locates and directly counts occurrences (as against counting the number of windows in which episodes occur), it facilitates the discovery of patterns with extra constraints (like being able to discover rules of the form "if A and B occur within 10 seconds of one another, C follows within another 20 seconds"). Another frequency measure was proposed in (Casas-Garriga 2003) where the user chooses the maximum inter-node distance allowed (instead of the window width which was needed earlier) and the algorithm automatically adjusts the window width based on the length of the episodes being counted. In (Laxman *et al* 2004b, 2005), two new frequency counts (referred to as the non-overlapped occurrence count and the non-interleaved occurrence count) are proposed based on directly counting some suitable subset of occurrences of episodes. These two counts (which are also automata-based counting schemes) have the same space complexity as the windows-based count of Mannila *et al* (1997) (i. e. l automata per episode for l-node episodes) but exhibit a significant advantage in terms of run-time efficiency. Moreover, the non-overlapped occurrences count is also theoretically elegant since it facilitates a connection between frequent episode discovery process and HMM learning (Laxman *et al* 2005). We will return to this aspect later in Sec. 5..

Graph-theoretic approaches have also been explored to locate episode occurrences in a sequence (Baeza-Yates 1991; Tronicek 2001; Hirao *et al* 2001). These algorithms, however, are more suited for search and retrieve applications rather than for discovery of *all* frequent episodes. The central idea here is to employ a preprocessing step to build a finite automaton called the DASG (Directed Acyclic Subsequence Graph), which accepts a string if and only if it is a subsequence of the given input sequence. It is possible to build this DASG for a sequence of length n in $\mathcal{O}(n\sigma)$ time, where σ is the size of the alphabet. Once the DASG is constructed, an episode of length l can be located in the sequence in linear, i. e. $\mathcal{O}(l)$, time.

4.3 *Patterns with explicit time constraints*

So far, we have discussed two major temporal pattern discovery frameworks in the form of sequential patterns (Agrawal & Srikant 1995) and frequent episodes (Mannila *et al* 1997). There is often a need for incorporating some time constraints into the structure of these temporal patterns. For example, the window width constraints (in both the windows-based as well as the minimal occurrences-based counting procedures) in frequent episode discovery are useful first-level timing information introduced into the patterns being discovered. In (Casas-Garriga 2003; Meger & Rigotti 2004), a maximum allowed inter-node time constraint (referred to as a maximum gap constraint) is used to dynamically alter window widths based on the lengths of episodes being discovered. Similarly, episode inter-node and expiry time constraints may be incorporated in the non-overlapped and non-interleaved occurrences-based counts (Laxman *et al* 2004b). In case of the sequential patterns framework (Srikanth & Agrawal 1996) proposed some generalizations to incorporate minimum and maximum time gap constraints between successive elements of a sequential pattern. Another interesting way to address inter-event time constraints is described by Bettini *et al* (1998). Here, multiple granularities (like hours, days, weeks etc.) are defined on the time axis and these are used to constrain the time between events in a temporal pattern. Timed finite automata (which were originally introduced in the context of modelling real time systems (Alur & Dill 1994)) are extended to the case where the transitions are governed by (in addition to the input symbol) the values associated with a set of clocks (which may be running in different granularities). These are referred to as *timed finite automata with granularities* (or TAGs) and are used to recognize frequent occurrences of the specialized temporal patterns.

In many temporal data mining scenarios, there is a need to incorporate timing information more explicitly into the patterns. This would give the patterns (and the rules generated from them) greater descriptive and inferential power. All techniques mentioned above treat events in the sequence as instantaneous. However, in many applications different events persist for different amounts of time and the durations of events carry important information. For example, in the case of the manufacturing plant data described earlier, the durations for which faults persist is important while trying to unearth hidden correlations among fault occurrences. Hence it is desirable to have a formulation for episodes where durations of events are incorporated. A framework that would facilitate description of such patterns, by incorporating event *dwelling time* constraints into the episode description is described by Laxman *et al* (2002). A similar idea in the context of sequential pattern mining (of, say, publication databases) is proposed by Lee *et al* (2003) where each item in a transaction is associated with an *exhibition time*.

Another useful timing information for temporal patterns is periodicity. Periodicity detection has been a much researched problem in signal processing for many years. For example, there are many applications that require the detection and tracking of the principal harmonic (which is closely related to the perceptual notion of pitch) in speech and other audio signals. Standard Fourier and autocorrelation analysis-based methods form the basis of most periodicity detection techniques that are currently in use in signal processing. In this review we focus on periodicity analysis techniques applicable for symbolic data streams with more of a data mining flavor.

The idea of *cyclic* association rules was introduced by Ozden *et al* (1998). The time axis is broken down into equally spaced user-defined time intervals and association rules of the variety used by Agrawal & Srikant 1994) that hold for the transactions in each of these time intervals are considered. An association rule is said to be cyclic if it holds with a fixed periodicity along the entire length of the sequence of time intervals. The task now is to obtain all cyclic association rules in a given database of time-stamped transactions. A straightforward approach to discovering such cyclic rules is presented by Ozden *et al* (1998). First, association rules in each time interval are generated using any standard association rule mining method. For each rule, the time intervals in which the rule holds is coded into a binary sequence and then the periodicity, if any, is detected in it to determine if the rule is cyclic or not.

The main difficulty in this approach is that it looks for patterns with *exact* periodicities. Just like in periodicity analysis for signal processing applications, in data mining also, there is a need to find some interesting ways to relax the periodicity constraints. One way to do this is by defining what can be called *partial* periodic patterns (Han *et al* 1999). Stated informally, a partial periodic pattern is a periodic pattern with wild cards or "don't cares" for some of its elements. For example, $A * B$, where $*$ denotes a wild card (i. e. any symbol from the alphabet), is a partial periodic pattern (of *time period* equal to 4 and length equal to 2) in the sequence $\langle ACBDABBQAWBX \rangle$. A further relaxation of the periodicity constraint can be incorporated by allowing for a few *misses* or skips in the occurrences of the pattern, so that not all, but typically *most* periods contain an occurrence of the pattern. Such situations are handled by Han *et al* (1999) by defining a *confidence* for the pattern. For example, the confidence, of a (partial) periodic pattern of period p is defined as the fraction of all periods of length p in the given data sequence (of which there are $\lfloor n/p \rfloor$ in a data sequence of length n) which match the pattern. A pattern that passes such a confidence constraint is sometimes referred to as a frequent periodic pattern. The discovery problem is now defined as follows: Given a sequence of events, a user-defined time period and a confidence threshold, find the complete set of frequent (partial) periodic patterns. Since, all sub-patterns of a frequent periodic pattern are also frequent and periodic (with the same time period) an Apriori-style algorithm is used to carry out this discovery task by first obtaining 1-length periodic patterns with the desired time period and then progressively growing these patterns to larger lengths.

Although this is quite an interesting algorithm for discovering partial periodic patterns, it is pointed out by Han *et al* (1999) that the Apriori property is not quite as effective for mining partial periodic patterns as it is for mining standard association rules. This is because, unlike in association rule mining, where the number of frequent k-itemsets falls quickly as k increases, in partial periodic patterns mining, the number of frequent k-patterns shrinks slowly with increasing k (due to a strong correlation between frequencies of patterns and their sub-patterns). Based on what they call the maximal sub-pattern hit set property, a novel data structure is proposed by Han *et al* (1998), which facilitates a more efficient partial pattern mining solution than the earlier Apriori-style counting algorithm.

The algorithms described by Han *et al* (1998, 1999) require the user to specify either one or a set of desired pattern time periods. Often potential time periods may vary over a wide range and it would be computationally infeasible to exhaustively try all meaningful time periods one after another. This issue can be addressed by first discovering all frequent time periods in the data and then proceeding with partial periodic pattern discovery for these time periods. Berberidis *et al* (2002) compute, for example, a circular autocorrelation function (using the FFT) to obtain a conservative set of candidate time periods for every symbol in the alphabet. Then the maximal sub-pattern tree method of Han *et al* (1998) is used to mine for periodic patterns given the set of candidate time periods so obtained. Another method to automatically obtain frequent time periods is proposed by Cao *et al* (2004). Here, for each symbol in each period in the sequence, the period position and frequency information is computed (in a single scan through the sequence) and stored in a 3-dimensional table called the Abbreviated List Table. Then, the frequent time periods in the data and their associated frequent periodic 1-patterns are obtained by analysing this table. These frequent periodic 1-patterns are used to grow the maximal sub-pattern tree for mining all partial periodic patterns.

The two main forms of periodicity constraint relaxations that have been considered so far are: (i) some elements of the patterns may be specified as wild cards, and (ii) periodicity may be occasionally disturbed through some “misses” or skips in the sequence of pattern occurrences. There are situations that might need other kinds of temporal disturbances to be tolerated in the periodicity definition. For example, a pattern’s periodicity may not persist for entire length of the sequence and so may manifest only in some (albeit sufficiently long) segment(s). Another case could be the need for allowing some lack of synchronization (and not just entire misses) in the sequence of pattern occurrences. This happens when some random noise events gets inserted in between a periodic sequence. These relaxations present many new challenges for automatic discovery of all partial periodic patterns of interest.

Ma & Hellerstein (2001) define a *p-pattern* which generalizes the idea of partial periodic patterns by incorporating some explicit time tolerances to account for such extra periodicity imperfections. As discussed in some earlier cases, here also, it is useful to automatically find potential time periods for these patterns. A chi-squared test-based approach is proposed to determine whether or not a candidate time period is a potentially frequent one for the given data, by comparing the number of inter-arrival times in the data with that time period against that for a random sequence of intervals. Another, more recent, generalization of the partial periodic patterns idea (for allowing additional tolerances in periodicity) is proposed by Cao *et al* (2004). Here, the user defines two thresholds, namely the minimum number of pattern repetitions and the maximum length of noise insertions between contiguous periodic pattern segments. A distance-based pruning method is presented to determine potential frequent time periods and some level-wise algorithms are described that can locate the longest partially periodic subsequence of the data corresponding to the frequent patterns associated with these time periods.

5. Statistical analysis of patterns in the data

From the preceding sections, it is clear that there is a wide variety of patterns which are of interest in temporal data-mining activity. Many efficient methods are available for matching and discovery of these patterns in large data sets. These techniques typically rely on the use of intelligent data structures and specialized counting algorithms to render them computationally feasible in the data mining context. One issue that has not been addressed, however, is the significance of the patterns so discovered. For example, a frequent episode is one whose frequency exceeds a user-defined threshold. But, how does the user know what thresholds to try? When can we say a pattern discovered in the data is significant (or interesting)? Given two patterns that were discovered in the data, is it possible to somehow quantify (in some statistical sense) the importance or relevance of one pattern over another? Is it possible to come up with some parameterless temporal pattern mining algorithms? Some recent work in temporal data mining research has been motivated by such considerations and we briefly explain these in this section.

5.1 *Significant episodes using Bernoulli or Markov models*

In order to determine when a pattern discovered in the data is significant, one broad class of approaches is as follows. We assume an underlying statistical model for the data. The parameters of the model can be estimated from some training data. With the model parameters known, one can determine (or approximate) the expected number of occurrences of a particular pattern in the data. Following this, if the number of times a pattern actually occurs in the given data deviates much from this expected value, then it is indicative of some unusual activity (and thus the pattern discovered is regarded as significant). Further, since the statistics governing the data generation process are known, it is possible to precisely quantify, for a given allowed probability of error, the extent of deviation (from the expected value) needed in order to call the pattern significant.

This general approach to statistical analysis of patterns, is largely based on some results in the context of determining the number of string occurrences in random text. For example, Bender & Kochman (1993), Regnier & Szpankowski (1998) show that if a Bernoulli or Markov assumption can be made on a text sequence, then the number of occurrences of a string in the sequence obeys the Central Limit Theorem. Similarly motivated approaches exist in the domain of computational biology as well. For instance, Pevzner *et al* (1989) consider patterns that allow fixed length gaps and determines the statistics of the number of occurrences of such patterns in random text. Flajolet *et al* (2001), extend these ideas to the case of patterns with arbitrary length gaps to address the intrusion detection problem in computer security.

An application of this general idea to the frequent episode discovery problem in temporal data mining is presented by Gwadera *et al* (2003). Under a Bernoulli model assumption, it is shown that the number of sliding windows over the data in which a given episode occurs at least once (i. e. the episode's frequency as defined by Mannila *et al* 1997), converges in distribution to a normal distribution with mean and variance determinable from the parameters of the underlying Bernoulli distribution (which are in turn estimated from some training data). Now, for a given user-defined confidence level, upper and lower thresholds for the observed frequency of an episode can be determined, using which, it is possible to call whether an episode is overrepresented or underrepresented (respectively) in the data. These ideas are extended by Atallah *et al* (2004) to the case of determining significance for a *set* of frequent episodes, and by Gwadera *et al* (2005), to the case of a Markov model assumption on the data sequence.

5.2 *Motif discovery under Markov assumption*

Another interesting statistical analysis of sequential patterns, with particular application to "motif" discovery in biological sequences is reported by Chudova & Smyth (2002). This analysis does not give us a significance testing framework for discovered patterns like was the case with the approach in § 5.1. Nevertheless, it provides a way to precisely quantify and assess the level of difficulty associated with the task of unearthing motif-like patterns in data (using a Markov assumption on the data). The analysis also provides a theoretical benchmark against which the performance of various motif discovery algorithms can be compared.

A simple pattern structure for motifs (which is known to be useful in computational biology) is considered, namely that of a "fixed-length plus noise" (Sze *et al* 2002). To model the embedding of a pattern, say $(P_1 P_2 \ldots P_L)$, in some background noise, a hidden Markov model (HMM) with L pattern states and one background state is considered. The ith pattern state emits P_i with a high probability of $(1 - \epsilon)$ where ϵ is the probability of substitution error. The background states can emit all symbols with equal probability. A "linear" state transition matrix is imposed on the states, i. e., ith pattern state transits only to the $(i + 1)$th pattern state, except the last one which transits only to the background state. The background state can make transitions either to itself or to the first pattern state. While, using such a structure implies that two occurrences of the pattern can only differ in substitution errors, a more general model is also considered which allows insertions and deletions as well.

By regarding the pattern detection as a binary classification problem (of whether a location in the sequence belongs to the pattern or the background) the Bayes error rate is determined for the HMM structure defined above. Since the Bayes error rate is known to be a lower bound on the error rates of all classifiers, it indicates, in some sense, the level of difficulty in the underlying pattern discovery problem. An analysis of how factors such as alphabet size, pattern length, pattern frequency, pattern autocorrelation and substitution error probability affect the Bayes error rate is provided.

Chudova & Smyth (2002), compare the empirical error rates for various motif discovery algorithms (that are currently in common use today) against the true Bayes error rate on a set of simulation-based motif-finding problems. It is observed that the performance of these methods can be quite far away from the optimal (Bayes) performance, unless very large training data sets are available. On real motif discovery problems, due to high pattern ambiguity, the Bayes error rate itself is quite high, suggesting that motif discovery based on sequence information alone is a hard problem. As a consequence, additional information outside of the sequence (like protein structure or gene expression measurements) need to be incorporated in future motif discovery algorithms.

5.3 *Episode generating HMMs*

A different approach to assessing significance of episodes discovered in time series data is proposed by Laxman *et al* (2004a, 2005). A formal connection is established between the frequent episode discovery framework and the learning of a class of specialized HMMs called Episode Generating HMMs (or EGHs). While only the case of serial episodes of Mannila *et al* (1997) is discussed, it is possible to extend the results by Laxman *et al* (2004a) to general episodes as well. In order to establish the relationship between episodes and EGHs, the non-overlapped occurrences-based frequency measure (Laxman *et al* 2004b) is used instead of the usual window-based frequency of Mannila *et al* (1997).

An EGH is an HMM with a restrictive state emission and transition structure which can embed serial episodes (without any substitution errors) in some background iid noise. It is composed of two kinds of states: episode states and noise states (each of equal number, say N). An episode state can emit only one of the symbols in the alphabet (with probability one), while all noise states can emit all symbols with equal probability. With exactly one symbol associated with each episode state, the emission structure is fully specified by the set of symbols $(A_1, \ldots, A_N)$, where the ith episode state can only emit the symbol A_i. The transition structure is entirely specified through one parameter called the noise parameter η. All transitions into noise states have probability η and transitions into episode states have probability $(1 - \eta)$. Each episode state is associated with one noise state in such a way that, it can transit only to either that noise state or to the "next" episode state (with last episode state allowed to transit back to the first). There are no self transitions allowed in an episode state. A noise state however can transit either into itself or into the corresponding "next" episode state.

The main results of Laxman *et al* (2004a) is as follows. Every episode is uniquely associated with an EGH. Given two episode α and β, and the corresponding EGH's Λ_α and Λ_β, the probability of Λ_α generating the data stream is more than that for Λ_β if and only if the frequency of α is greater than that of β. Then the maximum likelihood estimate of an EGH given any data stream is the EGH associated with the most frequent episode that occurs in the data.

An important consequence of this episode-EGH association is that it gives rise to a likelihood ratio test to assess significance of episodes discovered in the data. To carry out the significance analysis, we do not need to explicitly estimate any model for the data; we only need the frequency of episode, length of the data stream, the alphabet size and the size of the episode. Another interesting aspect is that for any fixed level of type I error, the frequency needed for an episode to be regarded as significant is inversely proportional to the episode size. The fact that smaller episodes have higher frequency thresholds is also interesting because it can further improve the efficiency of candidate generation during the frequent episode discovery process. Also the statistical analysis helps us to automatically fix a frequency threshold for the episodes, thus giving rise to what may be termed as parameterless data mining.

6. Concluding remarks

Analysing large sequential data streams to unearth any hidden regularities is important in many applications ranging from finance to manufacturing processes to bioinformatics. In this article, we have provided an overview of temporal data mining techniques for such problems. We have pointed out how many traditional methods from time series modelling & control, and pattern recognition are relevant here. However, in most applications we have to deal with symbolic data and often the objective is to unearth interesting (local) patterns. Hence the emphasis had been on techniques useful in such applications. We have considered in some detail, methods for discovering sequential patterns, frequent episodes and partial periodic patterns. We have also discussed some results regarding statistical analysis of such techniques.

Due to the increasing computerization in many fields, these days vast amounts of data are routinely collected. There is need for different kinds frameworks for unearthing useful knowledge that can be extracted from such databases. The field of temporal data mining is relatively young and one expects to see many new developments in the near future. In all data mining applications, the primary constraint is the large volume of data. Hence there is always a need for efficient algorithms. Improving time and space complexities of algorithms is a problem that would continue to attract attention. Another important issue is that of analysis of these algorithms so that one can assess the significance of the extracted patterns or rules in some statistical sense. Apart from this, there are many other interesting problems in temporal data mining that need to be addressed. We point out a couple of such issues below.

One important issue is that of what constitutes an interesting pattern in data. The notions of sequential patterns or frequent episodes represent only the currently popular structures for patterns. Experience with different applications would give rise to other useful notions and the problem of defining other structures for interesting patterns would be a problem that deserves attention. Another interesting problem is that of linking pattern discovery methods with those that estimate models for data generation process. For example, there are methods for learning mixture models from time series data (discussed in § 3.3). It is possible to learn such stochastic models (e.g., HMMs) for symbolic data also. On the other hand, given an event stream, we can find interesting patterns in the form of frequent episodes. While we have discussed some results to link such patterns with learning models in the form of HMMs, the problem of linking, in general, pattern discovery and learning of stochastic models for the data, is very much open. Such models can be very useful for better understanding of the underlying processes. One way to address this problem is reported by Mannila & Rusakov (2001) where under a stationarity and quasi-Markovian assumption, the event sequence is decomposed into independent components. The problem with this approach is that each event type is assumed to be emitted from only one of the sources in the mixture. Another approach is to use a mixture of Markov chains to model the data (Cadez *et al* 2000). It would be interesting to extend these ideas in order to build more sophisticated models such as mixture of HMMs. Learning such models in an unsupervised mode may be very difficult. Moreover, in a data mining context, we also need such learning algorithms to be very efficient in terms of both space and time. Another problem that has not received enough attention in temporal data mining is duration modelling for events in the sequence. As we have discussed, when different events have different durations, one can extend the basic framework of temporal patterns to define structures that allow for this and there are algorithms to discover frequent patters in this extended framework. However, there is little work in developing generative models for such data streams. Under a Markovian assumption, the dwelling times in any state would have distributions with memoryless property and hence accommodating arbitrary intervals for dwelling times is difficult. Hidden semi-Markov models have been proposed that relax the Markovian constraint to allow explicit modelling of dwelling times in the states (Rabiner 1989). Such modelling however significantly reduces the efficiency of the standard HMM learning algorithms. There is thus a need to find more efficient ways to incorporate duration modelling in HMM type models.

References

Agrawal R, Srikant R 1994 Fast algorithms for mining association rules in large databases. In *Proc. 20th Int. Conf. on Very Large Data Bases*, pp 487–499

Agrawal R, Srikant R 1995 Mining sequential patterns. In *Proc. 11th Int. Conf. on Data Engineering*, (Washington, DC: IEEE Comput. Soc.)

Agrawal R, Imielinski T, Swami 1993 A Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Conf. on Management of Data*, pp 207–216

Agrawal R, Lin K I, Sawhney H S, Shim K 1995a Fast similarity search in the presence of noise, scaling and translation in time series databases. In *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB95)*, pp 490–501

Agrawal R, Psaila G, Wimmers E L, Zait M 1995b Querying shapes of histories. In *Proc. 21st Int. Conf. on Very Large Databases*, Zurich, Switzerland

Alon J, Sclaroff S, Kollios G, Pavlovic V 2003 Discovering clusters in motion time series data. In *Proc. 2003 IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition*, pp I–375–I–381, Madison, Wisconsin

Alur R, Dill D L 1994 A theory of timed automata. *Theor. Comput. Sci.* 126: 183–235

Atallah M J, Gwadera R, Szpankowski W 2004 Detection of significant sets of episodes in event sequences. In *Proc. 4th IEEE Int. Conf. on Data Mining (ICDM 2004)*, pp 3–10, Brighton, UK

Baeza-Yates R A 1991 Searching subsequences. *Theor. Comput. Sci.* 78: 363–376

Baldi P, Chauvin Y, Hunkapiller T, McClure M 1994 Hidden Markov models of biological primary sequence information. *Proc. Nat. Acad. Sci. USA* 91: 1059–1063

Bender E A, Kochman F 1993 The distribution of subword counts is usually normal. *Eur. J. Combinatorics* 14: 265–275

Berberidis C, Vlahavas I P, Aref W G, Atallah M J, Elmagarmid A K 2002 On the discovery of weak periodicities in large time series. In *Lecture notes in computer science, Proc. 6th Eur. Conf. on Principles of Data Mining and Knowledge Discovery*, vol. 2431, pp 51–61

Bettini C, Wang X S, Jajodia S, Lin J L 1998 Discovering frequent event patterns with multiple granularities in time sequences. *IEEE Trans. Knowledge Data Eng.* 10: 222–237

Box G E P, Jenkins G M, Reinsel G C 1994 *Time series analysis: Forecasting and control* (Singapore: Pearson Education Inc.)

Cadez I, Heckerman D, Meek C, Smyth P, White S 2000 Model-based clustering and visualisation of navigation patterns on a web site. Technical Report CA 92717-3425, Dept. of Information and Computer Science, University of California, Irvine, CA

Cao H, Cheung D W, Mamoulis N 2004 Discovering partial periodic patterns in discrete data sequences. In *Proc. 8th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'04)*, Sydney, pp 653–658

Casas-Garriga G 2003 Discovering unbounded episodes in sequential data. In *Proc. 7th Eur. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, Cavtat-Dubvrovnik, Croatia, pp 83–94

Chang S F, Chen W, Men J, Sundaram H, Zhong D 1998 A fully automated content based video search engine supporting spatio-temporal queries. *IEEE Trans. Circuits Syst. Video Technol.* 8(5): 602–615

Chatfield C 1996 *The analysis of time series* 5th edn (New York, NY: Chapman and Hall)

Chudova D, Smyth P 2002 Pattern discovery in sequences under a Markovian assumption. In *Proc. Eigth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada

Cohen J 2004 Bioinformatics – an introduction for computer scientists. *ACM Comput. Surv.* 36(2): 122–158

Corpet F 1988 Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, 16: 10881–10890

Darrell T, Pentland A 1993 Space-time gestures. In *Proc. 1993 IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition (CVPR'93)*, pp 335–340

Dietterich T G, Michalski R S 1985 Discovering patterns in sequences of events. *Artif. Intell.* 25: 187–232

Duda R O, Hart P E, Stork D G 1997 *Pattern classification and scene analysis* (New York: Wiley)

Durbin R, Eddy S, Krogh A, Mitchison G 1998 *Biological sequence analysis* (Cambridge: University Press)

Ewens W J, Grant G R 2001 *Statistical methods in bioinformatics: An introduction* (New York: Springer-Verlag)

Fadili M J, Ruan S, Bloyet D, Mazoyer B 2000 A multistep unsupervised fuzzy clustering analysis of fMRI time series. *Human Brain Mapping* 10: 160–178

Flajolet P, Guivarc'h Y, Szpankowski W, Vallee B 2001 Hidden pattern statistics. In *Lecture notes in computer science; Proc. 28th Int. Colloq. on Automata, Languages and Programming* (London: Springer-Verlag) vol. 2076, pp 152–165

Frenkel K A 1991 The human genome project and informatics. *Commun. ACM* 34(11): 40–51

Garofalakis M, Rastogi R, Shim K 2002 Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowledge Data Eng.* 14: 530–552

Ghias A, Logan J, Chamberlin D, Smith B C 1995 Query by humming – musical information retrieval in an audio database. In *Proc. ACM Multimedia 95*, San Fransisco, CA

Gold B, Morgan N 2000 *Speech and audio signal processing: Processing and perception of speech and music* (New York: John Wiley & Sons)

Gray R M, Buzo A, Gray Jr. A H, Matsuyama Y 1980 Distortion measures for speech processing. *IEEE Trans. Acoust., Speech Signal Process.* 28: 367–376

Gusfield D 1997 *Algorithms on strings, trees and subsequences* (New York: University of Cambridge Press)

Gwadera R, Atallah M J, Szpankowski W 2003 Reliable detection of episodes in event sequences. In *Proc. 3rd IEEE Int. Conf. on Data Mining (ICDM 2003)*, pp 67–74

Gwadera R, Atallah M J, Szpankowski W 2005 Markov models for identification of significant episodes. In *Proc. 2005 SIAM Int. Conf. on Data Mining (SDM-05)*, Newport Beach, California

Han J, Kamber M 2001 *Data mining: Concepts and techniques* (San Fransisco, CA: Morgan Kauffmann)

Han J, Gong W, Yin Y 1998 Mining segment-wise periodic patterns in time-related databases. In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98)*, New York, pp 214–218

Han J, Dong G, Yin Y 1999 Efficient mining of partial periodic patterns in time series database. In *Proc. 15th Int. Conf. on Data Engineering, (ICDE'99)*, Sydney, pp 106–115

Hand D, Mannila H, Smyth P 2001 *Principles of data mining* (Cambridge, MA: MIT Press)

Haselsteiner E, Pfurtscheller G 2000 Using time-dependent neural networks for EEG classification. *IEEE Trans. Rehab. Eng.* 8: 457–463

Hastie T, Tibshirani R, Friedman J 2001 *The elements of statistical learning: Data mining, inference and prediction* (New York: Springer-Verlag)

Haykin S 1992 *Neural networks: A comprehensive foundation* (New York: Macmillan)

Hirao M, Inenaga S, Shinohara A, Takeda M, Arikawa S 2001 A practical algorithm to find the best episode patterns. *Lecture notes in computer science; Proc. 4th Int. Conf. on Discovery Science (DS 2001)* Washington, DC, vol. 2226, pp 435–441, 25–28

Juang B H, Rabiner L 1993 *Fundamentals of speech recognition.* (Englewood Cliffs, NJ: Prentice Hall)

Kalpakis K, Puttagunta D G V 2001 Distance measures for effective clustering of ARIMA time series. In *2001 IEEE Int. Conf. on Data Mining (ICDM01)*, San Jose, CA

Keogh E J, Pazzani M J 2000 Scaling up dynamic time warping for datamining applications. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining*, Boston, MA, pp 285–289, 20–23

Koskela T, Lehtokangas M, Saarinen J, Kaski K 1996 Time series prediction with multilayer perceptron, FIR and Elman neural networks. In *Proc. World Congress on Neural Networks*, pp 491–496

Kruskal J B 1983 An overview of sequence comparison: Time warps, string edits and macromolecules. *SIAM Rev.* 21: 201–237

Kundu A, He Y, Bahl P 1988 Word recognition and word hypothesis generation for handwritten script: A Hidden Markov Model based approach. In *Proc. 1988 IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition (CVPR'88)*, pp 457–462

Law M H, Kwok J T 2000 Rival penalized competitive learning for model-based sequence clustering. In *Proc. IEEE Int. Conf. on Pattern Recognition (ICPR00)*, Barcelona, Spain

Laxman S, Sastry P S, Unnikrishnan K P 2002 Generalized frequent episodes in event sequences. *Temporal Data Mining Workshop Notes, SIGKDD*, (eds) K P Unnikrishnan, R Uthurusamy, Edmonton, Alberta, Canada

Laxman S, Sastry P S, Unnikrishnan K P 2004a Fast algorithms for frequent episode discovery in event sequences. Technical Report CL-2004-04/MSR, GM R&D Center, Warren

Laxman S, Sastry P S, Unnikrishnan K P 2004b Fast algorithms for frequent episode discovery in event sequences. In *Proc. 3rd Workshop on Mining Temporal and Sequential Data*, Seattle, WA

Laxman S, Sastry P S, Unnikrishnan K P 2005 Discovering frequent episodes and learning hidden markov models: A formal connection. *IEEE Trans. Knowledge Data Eng.* 17: 1505–1517

Lee C-H, Chen M-S, Lin C-R 2003 Progressive pattern miner: An efficient algorithm for mining general temporal association rules. *IEEE Trans. Knowledge Data Eng.* 15: 1004–1017

Levenshtein V I 1966 Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.* 10: 707–710

Lin M-Y, Lee S-Y 2003 Improving the efficiency of interactive sequential pattern mining by incremental pattern discovery. In *Proc. IEEE 36th Annu. Hawaii Int. Conf. on System Sciences (HICSS03)*, Big Island, Hawaii

Ma S, Hellerstein J L 2001 Mining partially periodic event patterns with unknown periods. In *Proc. 17th Int. Conf. on Data Eng. (ICDE'01)*, pp 205–214

Mannila H, Rusakov D 2001 Decomposition of event sequences into independent components. In *First SIAM Int. Conf. on Data Mining*, Chicago, IL

Mannila H, Toivonen H, Verkamo A I 1997 Discovery of frequent episodes in event sequences. *Data Mining Knowledge Discovery* 1: 259–289

Meger N, Rigotti C 2004 Constraint-based mining of episode rules and optimal window sizes. In *Proc. 8th Eur. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, Pisa, Italy

Miller R T, Christoffels A G, Gopalakrishnan C, Burke J, Ptitsyn A A, Broveak T R, Hide W A 1999 A comprehensive approach to clustering of expressed human gene sequence: The sequence tag alignment and consensus knowledge base. *Genome Res.* 9: 1143–1155

Miller W, Schwartz S, Hardison R C 1994 A point of contact between computer science and molecular biology. *IEEE Comput. Sci. Eng.* 1: 69–78

Nag R, Wong K H, Fallside F 1986 Script recognition using Hidden Markov Models. In *Proc. 1986 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'86)*, pp 2071–2074

Nalwa V S 1997 Automatic on-line signature verification. *Proc. IEEE* 85: 215–239

Ng R T, Lakshmanan L V S, Han J, Pang A 1998 Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM SIGMOD Int. Conf. on Management of Data*, (Seattle, Washington) pp 13–24

Oates T, Firoiu L, Cohen P R 2001 Using dynamic time warping to bootstrap HMM-based clustering of time series. In *Lecture notes in computer science; Sequence learning: Paradigms, algorithms, and applications* (eds) C L Giles, R Sun (Heidelberg: Springer-Verlag) vol. 1828, p. 35

Osata N *et al* 2002 A computer-based method of selecting clones for a full-length cDNA project: Simulataneous collection of negligibly redundant and variant cDNAs. *Genome Res.* 12: 1127–1134

O'Shaughnessy D 2003 *Speech communications: Human and machine* (Piscataway, NJ: IEEE Press)

Ozden B, Ramaswamy S, Silberschatz A 1998 Cyclic association rules. In *Proc. 14th Int. Conf. on Data Engineering (ICDE'98)*, Orlando, Florida, pp 412–421

Pasquier N, Bastide Y, Taouil R, Lakhal L 1999 Discovering frequent closed itemsets for association rules. In *Lecture notes in computer science; Proc. 7th Int. Conf. on Database Theory (ICDT99)*, Jerusalem, Israel, vol. 1540, pp 398–416

Perng C-S, Wang H, Zhang S R, Parker D S 2000 Landmarks: A new model for similarity-based pattern querying in time series databases. In *16th Int. Conf. on Data Engineering (ICDE00)*, p. 33, San Diego, CA

Pevzner P A, Borodovski M Y, Mironov A A 1989 Linguistic of nucleotide sequences: The significance of deviation from mean statistical characteristics and prediction of the frequencies of occurrence of words. *J. Biomol. Struct. Dynamics* 6: 1013–1026

Rabiner L R 1989 A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77: 257–286

Regnier M, Szpankowski W 1998 On pattern frequency occurrences in a Markovian sequence. *Algorithmica* 22: 631–649

Schreiber T, Schmitz A 1997 Classification of time series data with nonlinear similarity measures. *Phys. Rev. Lett.* 79: 1475–1478

Sclaroff S, Kollios G, Betke M, Rosales R 2001 Motion mining. In *Lecture notes in computer science; Proc. 2nd Int. Workshop on Multimedia Databases and Image Communication* (Heidelberg: Springer-Verlag)

Sebastiani P, Ramoni M, Cohen P, Warwick J, Davis J 1999 Discovering dynamics using bayesian clustering. In *Lecture notes in computer science; Adv. in Intelligent Data Analysis: 3rd Int. Symp., IDA-99* (Heidelberg: Springer-Verlag) vol. 1642, p. 199

Shintani T, Kitsuregawa M 1998 Mining algorithms for sequential patterns in parallel: Hash based approach. In *Proc. 2nd Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pp 283–294

Smyth P 1997 Clustering sequences with hidden Markov models. *Adv. Neural Inf. Process.* 9: 648–655

Smyth P 2001 Data mining at the interface of computer science and statistics. In *Data mining for scientific and engineering applications.* (eds) R L Grossman, C Kamath, P Kegelmeyer, V Kumar, R R Namburu (Dordrecht: Kluwer Academic)

Srikanth R, Agrawal R 1996 Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. on Extending Database Technology (EDBT)*, Avignon, France

Starner T E, Pentland A 1995 Visual recognition of American sign language. In *Proc. 1995 Int. Workshop on Face and Gesture Recognition*, Zurich

Sutton R S 1988 Learning to predict by method of temporal differences. *Machine Learning* 3(1): 9–44

Sze S H, Gelfand M S, Pevzner P A 2002 Finding weak motifs in DNA sequences. In *Proc. 2002 Pacific Symposium on Biocomputing*, pp 235–246

Tappert C C, Suen C Y, Wakahara T 1990 The state of the art in on-line handwriting recognition. *IEEE Trans. Pattern Anal. Machine Intell.* 12: 787–808

Tino P, Schittenkopf C, Dorffner G 2000 Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: Lessons learned from financial volatility trading (url:citeseer.nj.nec.com/tino00temporal.html)

Tronicek Z 2001 Episode matching. In *Proc. 12th Annu. Symp. on Combinatorial Pattern Matching (CPM 2001)*, Jerusalem, Israel, vol. 2089, pp 143–146

Wan E A 1990 Temporal backpropagation for FIR neural networks. In *Int. Joint Conf. on Neural Networks (1990 IJCNN)*, vol. 1, pp 575–580

Wang J T-L, Chirn G-W, Marr T G, Shapiro B, Shasha D, Zhang K 1994 Combinatorial pattern discovery for scientific data: some preliminary results. In *Proc. 1994 ACM SIGMOD Int. Conf. on Management of Data*, Minneapolis, Minnesota, pp 115–125

Wang J, Han J 2004 BIDE: Efficient mining of frequent closed sequences. In *20th Int. Conf. on Data Engineering*, Boston, MA

Witten I H, Frank E 2000 *Data mining: Practical machine learning tools and techniques with JAVA implementations* (San Fransisco, CA: Morgan Kaufmann)

Wu C, Berry M, Shivakumar S, McLarty J 1995 Neural networks for full-scale protein sequence classification: Sequence encoding with singular value decomposition. *Machine Learning*, Special issue on applications in molecular biology 21(1–2): 177–193

Wu S, Manber U 1992 Fast text searching allowing errors. *Commun. ACM* 35(10): 83–91

Wu Y-L, Agrawal D, Abbadi A E 2000 A comparison of DFT and DWT based similarity search in time series databases. In *Proc. Ninth Int. Conf. on Information and Knowledge Management*, McLean, VA, pp 488–495

Xiong Y, Yeung D Y 2002 Mixtures of ARMA models for model-based time series clustering. In *2002 IEEE Int. Conf. on Data Mining*, Maebashi City, Japan, pp 717–720

Yamato J, Ohya J, Ishii K 1992 Recognizing human action in time-sequential images using Hidden Markov Model. In *Proc. 1992 IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition (CVPR'92)*, Champaign, IL, pp 379–385

Yan X, Han J, Afshar R 2003 CloSpan: Mining closed sequential patterns in large datasets. In *Proc. 2003 Int. SIAM Conf. on Data Mining (SDM03)*, San Fransisco, CA

Yule G 1927 On a method of investigating periodicity in distributed series with special reference to Wolfer's sunspot numbers. *Philos. Trans. R. Soc. London* A226

Zaki M J 1998 Efficient enumeration of frequent sequences. In *Proc. ACM 7th Int. Conf. Information and Knowledge Management (CIKM)*