Necessary idea but no details on how the algorithm works

Finding Approximate Frequent Patterns in Streaming Medical Data

Jessica Lin
Computer Science Department
George Mason University
Fairfax, VA
jessica@cs.gmu.edu

Yuan Li
Computer Science Department
George Mason University
Fairfax, VA
ylif@gmu.edu

Abstract

Time series data is ubiquitous and plays an important role in virtually every domain. For example, in medicine, the advancement of computer technology has enabled more sophisticated patients monitoring, either on-site or remotely. Such monitoring produces massive amount of time series data, which contain valuable information for pattern learning and knowledge discovery. In this paper, we explore the problem of identifying frequently occurring patterns, or motifs, in streaming medical data. The problem of frequent patterns mining has many potential applications, including compression, summarization, and event prediction. We propose a novel approach based on grammar induction that allows the discovery of approximate, variable-length motifs in streaming data. The preliminary results show that the grammar-based approach is able to find some important motifs in some medical data, and suggest that using grammar-based algorithms for time series pattern discovery might be worth exploring.

Keywords

Time Series, Frequent Patterns, Grammar Induction

1. Introduction

Time series data is ubiquitous and plays an important role in virtually every domain. For example, in medicine, the advancement of computer technology has enabled more sophisticated patients monitoring, either on-site or remotely. With the massive amount of medical data produced everyday, it has become increasingly apparent that efficient methods to search and analyze historic data, and to detect events in real-time streaming data are in great demand. Examples of medical time series or streaming data include electrocardiogram (ECG) signals, respiration rates, blood pressure, etc.

The task of frequent pattern mining is an important problem that has many applications. In addition to its own merit of summarizing and compressing data, it is also a pre-cursor to association rule or sequential pattern mining [2]. The frequent patterns mined can potentially be helpful in predicting future events. For example, there has been some work on the extraction of significant patterns for heart attack prediction [38]. In bioinformatics, it is well understood that overrepresented DNA sequences often have biological significance [9, 11, 12, 28, 32]. A substantial body of literature has been devoted to techniques to discover such patterns [2, 3].

In a previous work, we defined the related concept of "time series motif" [18], which are frequently occurring patterns in time series data. Since then, a great deal of work has been proposed for the discovery of time series motifs [5, 7, 18, 20, 21, 22. 23, 29, 30, 31]. Figure 1 shows an example of a time series motif in an ECG dataset.

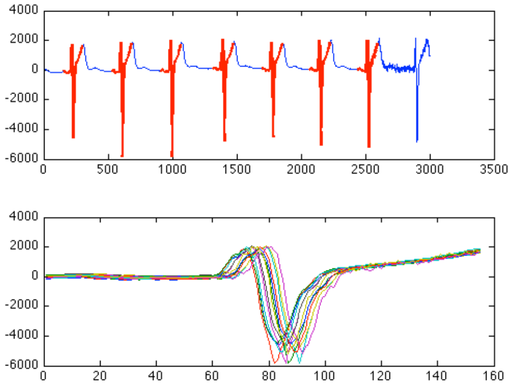

Figure 1. (Top) Original ECG time series. Matches for a motif of length 159 is highlighted. The initial length used is 150. (Bottom) The discovered motif instances are plotted.

Intuitively, one may try to convert a real-valued time series data into a data format for which existing off-the-shelf algorithms can be applied. One way to achieve this is to discretize each data value in the time series. However, the problem with this approach is that time series data are typically noisy, and considering every single point as an "event" would result in a noisy string that inaccurately reflects the noises as true patterns in the data. A better alternative is to consider subsequences instead. However, this approach poses another challenge: we simply cannot know in advance which subsequences are frequent. As a result, we need to consider *every* possible subsequence in the data. Considering all subsequences of any length, with overlaps, is undoubtedly a tedious task. To alleviate the complexity, most existing work [7, 18, 22, 23] thus require an input parameter: a pre-defined motif length *n*. This limits the search space for the algorithms; however, it also implies that the length of motifs (n) must be known in advance. In addition, frequent patterns of different lengths might co-exist within the same dataset. In order to find all significant patterns with unknown lengths, one would need to repeat the motif discovery algorithm several times—each time with a different window size. It is more desirable to have an algorithm that can automatically detect significant motifs of variable, previously unknown lengths, without exhaustively trying different subsequence lengths.

In this work, we propose to utilize a grammar-based compression algorithm, *Sequitur* [24], that can automatically and efficiently identify frequent patterns and hierarchical structure in data. We believe that a grammar-based approach [13, 14, 16, 24, 33] is suitable and advantageous for our task due to several reasons. First, producing a (relatively) small set of interpretable rules from a massive dataset is a desirable goal of data mining. Clearly, a grammar-based method will allow a more natural mapping from data to rules [34], and can reveal the hidden hierarchical structures in the data. There has also been increasing interest in grammar-based methods for feature extraction, classification and forecasting of time series [10, 36]. Furthermore, as mentioned earlier, it is important to consider every subsequence in the time series when trying to identify motifs.

While this work is still at its early stage, the preliminary results are promising. Specifically, they show that the grammar-based approach has the potential to identify some important motifs in time series.

The rest of the paper is organized as follows. Section 2 discusses background and related work on time series motifs and grammar induction. Section 3 describes the grammar induction algorithm, *Sequitur*, that we adapt in this work. We describe our approach in Section 4. Section 5 presents some preliminary results using grammar-based approach to find variable-length motifs. We conclude in Section 6 and discuss future work.

2. Background and Related Work

In this section, we briefly discuss background and related work on time series similarity search.

For concreteness, we begin with definitions of time series:

Definition 1. *Time Series*: A time series $T = t_1,\dots,t_m$ is an ordered set of m real-valued variables.

Since we are interested in finding local patterns, we consider time series subsequences as the basic unit:

Definition 2. *Subsequence:* Given a time series T of length m, a subsequence C of T is a subsection of length $n \leq m$ of contiguous position from p, that is, $C = t_p \dots t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.

Since all subsequences may potentially be the candidates for motifs, any algorithm would have to extract and consider all of them. This can be achieved via the use of a sliding window:

Definition 4. *Sliding Window:* Given a time series T and a user-defined subsequence length n, all possible subsequences can be extracted by sliding a window of size n across T and considering each subsequence C_p, for $1 \leq p \leq m - n + 1$.

Next, we briefly discuss related work on time series motif discovery and grammar induction techniques.

2.1 Related Work

In [18], we defined time series motif C as the subsequence in T that has the highest count of non-trivial matches, that is, subsequences that are within ε units of distance away from C. We proposed a sub-quadratic algorithm to find exact motifs of a given length. Mueen et al proposed an algorithm named MK that is an improvement from the brute-force exact motif discovery algorithm [23]. In some applications, it may be sufficient or even desirable to have a fast algorithm that can find *approximate* motifs [7]. As an example, Chiu and Keogh proposed probabilistic motif discovery algorithm based on random projection [7, 32]. The advantage of probabilistic motif discovery algorithm is its efficiency. Other approximate motifs algorithms exist [5, 27, 30, 35]; however, one common drawback for all these algorithms is that they require an input parameter for the motif length.

A few algorithms were proposed to discover motifs of variable lengths [21, 25, 30]; however, they either do so via post-processing, scale poorly, or quantize the whole data rather than considering overlapping subsequences, resulting in inaccurate and incomplete patterns found.

3. Sequitur

We investigate how to extract patterns using techniques that identify hierarchies and frequent sequences. Although aimed at compressing discrete sequences of data, there are algorithms that can be used as a proof-of-concept for our preliminary study. For instance, *Sequitur* [24] is a string compression algorithm that infers a context-free grammar from a sequence of discrete symbols [24]. It has been adopted in various domains due to the many nice properties it offers: it has been used to find repeated DNA sequences [6, 33] and repeated function call sequences [15], and to segment time series [4]. The main premise is that repeated subsequences are replaced by a grammatical rule that generates the subsequence, thereby reducing the length of the original sequence, and producing a hierarchical representation that summarizes the structure of the data. Although simple in design, *Sequitur* has been shown to be competitive with the state-of-the-art compression algorithms [24], maintaining its scalability even for large sequences. Moreover, *Sequitur* offers a unique advantage—it utilizes and identifies the hidden structure (recurring subsequences) in the input data sequence, requiring relatively small memory footprint. Due to these reasons, we choose *Sequitur* in this work to demonstrate the utility of using grammar-based compression algorithms to find patterns in time series data.

Sequitur works by maintaining two properties: digram uniqueness and rule utility [24]. The first property governs that no pair of consecutive symbols (terminals or non-terminals) can appear more than once. When *Sequitur* reads a new symbol from the input sequence, the last two symbols of the sequence read so far—the new symbol and its predecessor symbol—form a digram [24]. A table that stores all existing digrams is maintained. If this new digram already exists in the digram table, i.e., it appears somewhere in the sequence already read, *Sequitur* uses a non-terminal to substitute these digrams, and, if such rule[1] has not yet existed, it forms a new grammar rule with the non-terminal on the left hand side. The second property, rule uniqueness, ensures that each grammar rule be used more than once except for the top-level rule, since a grammar rule that occurs just once is not meaningful and should be removed. As an example, the input string S1: "12131213412" can be converted to the following grammar:

Grammar rule	*Expanded Grammar rule*
S1 -> BB4A	*12131213412*
A -> 12	*12*
B -> A13	*1213*

The top-level grammar rule, *S1→ BB4A*, denotes the sequence seen so far. *Sequitur* is an online algorithm that generates the grammar incrementally as each symbol arrives. It is, therefore, ideal for the streaming scenarios. It is both time- and space-efficient, requiring O(m) time to compress a sequence of size m, and a compressed sequence is of size O(m) in the worst case (i.e., no compression), and O(log$m$) in the best case [24].

The main advantages of *Sequitur* (or many grammar-induction algorithms in general) are three-fold: (1) it identifies recurring patterns automatically, e.g., "1213" in the previous example, as well as hierarchical structure; (2) the recurring patterns found can be of any lengths; and (3) it is suitable for streaming data since it constructs the grammars in an incremental fashion. These benefits suggest that we may be able to adapt it to find variable-length motifs for time series. We describe how we achieve this in the next section.

4. Finding Approximate Variable-Length Motifs By Sequitur

We propose an algorithm that finds approximate variable-length motifs using *Sequitur*. Our approach consists of three steps: Pre-processing (discretization), Motif Discovery (*Sequitur*), and Post-processing. We describe each step in more details below.

4.1 Step 1: Discretization

Sequitur, or more generally, grammar induction algorithms, were originally designed for discrete data. However, time series are real-valued data, requiring a pre-processing step to allow the application of a grammar-based algorithm.

In a previous work, we introduced a time series symbolic representation called Symbolic Aggregate approXimation (SAX) [1, 17, 19]. While there have been dozens of symbolic representations proposed for time series data, SAX has been shown to outperform existing methods. In addition, SAX has some unique, desirable properties such as dimensionality reduction, lower-bounding distance measures, and equiprobable symbols. For these reasons, we will utilize SAX for our pre-processing step.

Given a time series, SAX performs discretization by dividing the time series into w equal-sized segments. For each segment, their mean value is computed, and then mapped to a symbol according to a set of breakpoints that divide the distribution space into α equiprobable regions, where α is the alphabet size specified by the user. If the symbols were not equiprobable, some of the symbols would occur more frequently than others. As a consequence, we would inject a probabilistic bias in the process. It has been noted that some data structures such as suffix trees produce optimal results when the symbols are of equiprobability [8]. The discretization steps are summarized in Figure 2.

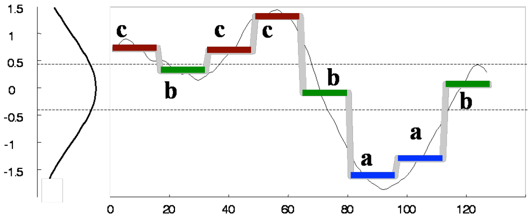

Figure 2. Example of SAX for a time series. The time series above is transformed to the string *cbccbaab*, and the dimensionality is reduced from 128 to 8.

Since each subsequence in the dataset can be a potential candidate for motif, we should consider all possible subsequences in order to ensure correct results. This can be achieved by using a sliding window of length n across the time series. Note that n is just the initial window length for our algorithm; the algorithm will grow the patterns automatically. Once we collect all the subsequences, we can then discretize each subsequence individually using SAX, and then concatenate them to form one single sequence. A transformed sequence might look

[1] Note the "rules" here should not be confused with the sequential or association rules. The "rules" here refer to those that are generated by the algorithm. It is equivalent to the concept of frequent itemsets in association rule mining.

something like this:

S = *1131-1132-1232-1223-1344-1131-1132-1232*...

where the '-' denotes the delimiter between consecutive, overlapping subsequences.

The reason we choose to discretize subsequences rather than individual points is that time series are typically very noisy. If we discretize each time point into a symbol, and then form a string from these symbols, then we would give equal weight to each time point, including the noises. On the other hand, the "aggregating" feature of SAX would smooth out the subsequences and essentially remove the noises.

4.2 Step 2: Sequitur on SAX Words

Once we transform the time series into a discrete sequence consisted of SAX strings, the application of *Sequitur* on the sequence is straight-forward. Each string delimited by '-' represents one subsequence, and is treated as a terminal symbol, an atomic unit for patterns. *Sequitur* embodies efficiency and accuracy in finding the repeated patterns of sequences in many cases. One possible grammar rule that can be generated from the above string is

$A \rightarrow$ 1131-1132-1232.

We modify the original *Sequitur* algorithm and record the offsets of the subsequences that occur in each grammar rule.

4.3 Step 3: Post-Processing

Since we discretized the data before running the algorithm, we now need to map the rules and frequent strings back to the time series subsequences. We can simply record the starting offsets of all grammar rule instances. The number of rules generated can be large and, similar to association rules mining [2], not all rules are interesting or important. Several refinement steps can be performed on the grammar rules. In this work, we performed the following refinement: eliminating trivial matches, ranking rules by their "interestingness" denoted by frequency, rule length, and pattern variation.

5. Empirical Evaluation

In this section we evaluate the potential of using *Sequitur* to find frequent patterns in medical time series data. While the experimental evaluation is brief, as this work is still at its early stage, the following examples show that *Sequitur* can find *some* time series motifs without knowing the exact lengths of the motifs in advance. For all examples shown below, we choose $\alpha = 4$ and $w = 4$, which are both arbitrary choices that have been shown to work well for most datasets [19].

As a sanity check, the first dataset we used is the ECG dataset shown in Figure 1 (see Section 1). The obvious motif, i.e. the individual heartbeats, are indeed discovered. The length of the motif shown is 159, with the initial window length of 150. Figure 3 below shows two instances of a motif found in a heart rate dataset from a patient monitored in Intensive Care Unit. The data is obtained from UCI Machine Learning Repository [39].

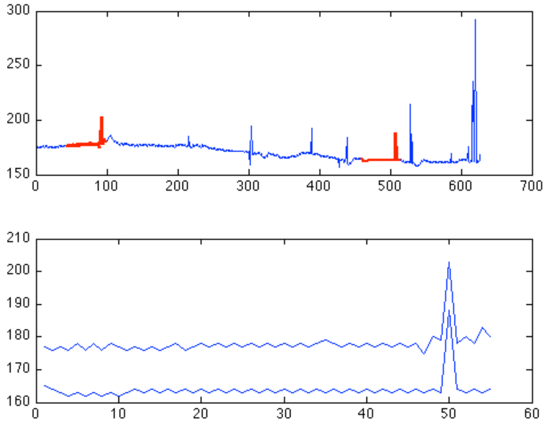

Figure 3. (Top) A patient's heart rate data. Matches for a motif of length 54 are highlighted. (Bottom) The discovered motif instances are plotted.

In Figure 4, we show a motif found in the winding dataset from UCR Time Series Archive [37]. A motif of length 84 is found, when the initial window length is 50.

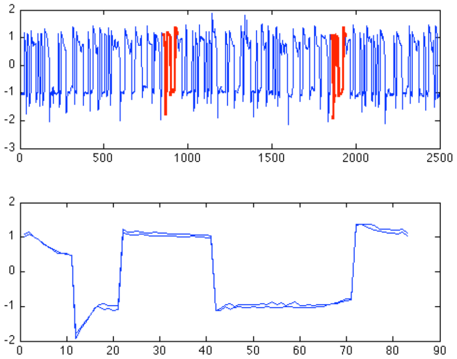

Figure 4. (Top) Original winding dataset. Matches for a motif of length 84 is highlighted. The initial length used is 50. (Bottom). The discovered motif instances are plotted.

6. Conclusion

In this preliminary work, we propose a methodology to find approximate variable-length medical time series motif using an efficient grammar-based compression algorithm. Existing algorithms that discover motifs of variable lengths either do so via post-processing, scale poorly, or quantize the whole data rather than considering overlapping subsequences, resulting in inaccurate and incomplete patterns found. Our algorithm mitigates the shortcomings;

in addition, it offers the advantage of discovering hierarchical structure, regularity and grammar from the data. The preliminary results are promising. They show that the grammar-based approach is able to find some important motifs in some medical data and suggest that using grammar-based algorithms for time series pattern discovery might be worth exploring. A natural question to ask now is, would our algorithm be useful for finding patterns or predicting future events in streaming medical data?

Several other future directions are possible. Due to the preliminary stage of this work, the literature comparison is non-existent. It is partly due to the fact that to the best of our knowledge, there is no known algorithm that can find variable-length frequent patterns with comparable efficiency. Nonetheless, we would like compare with some existing methods to further validate our findings. In addition, the post-processing step of the algorithm can be further refined for better motif identification. It would be useful to rank motifs based on their "interestingness." Furthermore, we would like to investigate the utilities of our algorithm on finding hierarchical structures and grammars from medical time series data.

7. References

1. SAX page: http://www.cs.gmu.edu/~jessica/sax.htm
2. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In proceedings of the 1993 ACM SIGMOD Int'l Conference on Management of Data. Washington, D.C. May 26-28, 1993. pp. 207-216
3. R. Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In Proc. of the 11th Int'l Conference on Data Engineering, Taipei, Taiwan, March 1995.
4. T. Armstrong and T .Oates. RIPTIDE: Segmenting Data Using Multiple Resolutions. In the Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL), 2007.
5. P. Beaudoin, M. van de Panne, P. Poulin and S. Coros, *Motion-Motif Graphs*, Symposium on Computer Animation 2008.
6. N. Cherniavsky and R. Ladner. Grammar-based Compression of DNA Sequences. UW CSE Technical Report 2007-05-02.
7. Chiu, B. Keogh, E., & Lonardi, S. (2003). Probabilistic Discovery of Time Series Motifs. In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 24 - 27, 2003. Washington, DC, USA. pp 493-498.
8. M. Crochemore, A. Czumaj, L. Gasjeniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter. Speeding Up Two String-Matching Algorithms. Algorithmica. vol. 12. pp. 247-267. 1994.
9. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids: Cambridge University Press. 1998.
10. D. Eads, E. Rosten, D. Helmbold. "Grammar-guided Feature Extraction for Location-Based Object Detection." British Machine Vision Conference. Queen Mary, University of London. London, UK. September 11, 2009.
11. A. Gionis and H. Mannila. Finding Recurrent Sources in Sequences. In proceedings of the 7th Int'l Conference on Research in Computational Molecular Biology. Berlin, Germany. 2003. pp. 123-130
12. D. He. Using Suffix Tree to Discover Complex Repetitive Patterns in DNA Sequences, The 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE EMBC 2006, New York City, New York, USA, August 30 - September 3, 2006
13. P. Langley. Simplicity and Representation Change in Grammar Induction. Technical Report. 1995.
14. P. Langley and S. Stromsten. Learning Context-Free Grammars with a Simplicity Bias. In proceedings of the 11th International Conference on Machine Learning. Standord, CA.
15. J.R. Larus. Whole program paths. SIGPLAN Not. 34, 5, pp. 259-269, 1999.
16. E. Lehman. Approximation Algorithms for Grammar-Based Data Compression, PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002.
17. J. Lin, E. Keogh, S. Lonardi, and B. Chiu, A Symbolic Representation of Time Series, with Implications for Streaming Algorithms, Workshop on Research Issues in Data Mining and Knowledge Discovery, the 8th ACM SIGMOD. San Diego, CA, 2003.
18. J. Lin, E. Keogh, P. Patel, and S. Lonardi, Finding Motifs in Time Series, the 2nd Workshop on Temporal Data Mining, the 8th ACM Int'l Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, 2002, pp. 53-68.
19. J. Lin, E. Keogh, W. Li, and S. Lonardi. (2007). Experiencing SAX: A Novel Symbolic Representation of Time Series. Data Mining and Knowledge Discovery Journal.
20. J. Meng, J.Yuan, M. Hans and Y. Wu, *Mining Motifs from Human Motion*, Proc. of EUROGRAPHICS, 2008.
21. D. Minnen, T. Starner, I. Essa, C. Isbell. Activity Discovery: Sparse Motifs from Multivariate Time Series. Snowbird Learning Workshop, Snowbird, Utah, April 4-7, 2006.
22. D. Minnen, C.L. Isbell, I. Essa, and T. Starner. Discovering Multivariate Motifs using Subsequence Density Estimation and Greedy Mixture Learning. Twenty-Second Conf. on Artificial Intelligence (AAAI-07), Vancouver, B.C., July 22-26, 2007.

23. A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact Discovery of Time Series Motifs. In proceedings of the 2009 SIAM International Conference on Data Mining (SDM09). April 30-May 2, 2009. Sparks, NV.
24. C.G. Nevill-Manning and I.H. Witten. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. Journal of Artificial Intelligence Research, 7, 67-82.
25. T. Oates. PERUSE: An Unsupervised Algorithm for Finding Recurring Patterns in Time Series. In proceedings of the International Conference on Data Mining. Maebashi City, Japan. Dec 9-12. pp. 330-337.
26. M. Riesenhuber and T. Poggio. Hierarchical Models of Object Recognition in Cortex. Nature Neuroscience 2: 1019:1025.
27. S. Rombo and G. Terracina, *Discovering representative models in large time series databases,* Proc. of the 6th International Conference on Flexible Query Answering Systems, pp. 84–97, 2004.
28. R. Staden. Methods for Discovering Novel Motifs in Nucleic Acid Sequences. Computer Applications in Biosciences. vol. 5. pp. 293-298. 1989.
29. Y. Tanaka and K. Uehara. Motif Discovery Algorithm from Motion Data. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence. Kanazawa, Japan. June 2-4, 2004.
30. Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. Mach. Learn. 58, 2-3 (Feb. 2005), 269-300.
31. H. Tang and S.S. Liao. Discovering original motifs with different lengths from time series. Know.-Based Syst. 21, 7 (Oct. 2008), 666-671.
32. M. Tompa and J. Buhler. Finding Motifs Using Random Projections. In proceedings of the 5th Int'l Conference on Computational Molecular Biology. Montreal, Canada. Apr 22-25, 2001. pp. 67-74
33. R. Ladner. Enhanced Sequitur for Finding Structure in Data. In Proceedings of the 2003 Data Compression Conference. March 25-27, Snowbird, UT. pp 425-.
34. M.L. Wong and K.S. Leung. Data mining using grammar based genetic programming and applications. In: Genetic programming, vol. 3. The Netherlands: Kluwer Academic Publishers; 2000.
35. T. Guyet, C. Garbay and M. Dojat, *Knowledge construction from time series data using a collaborative exploration system*, Journal of Biomedical Informatics 40(6): 672-687 (2007).
36. E. Keogh. Personal Communications.
37. E. Keogh The UCR Time Series Data Mining Archive. http://www.cs. ucr.edu/~eamonn/tsdma/index.html
38. S. Patil & Y. S. Kumaraswamy. Extraction of Significant Patterns from Heart Disease Warehouses for Heart Attack Prediction. International Journal of Computer Science and Network Security, Vol 9. No. 2. February, 2009.
39. Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.